TESIS - TE142599

PENGENDALIAN DISTRIBUSI *QUADROTOR SWARM* UNTUK PELACAKAN TITIK *CENTROID* DENGAN MENGGUNAKAN *MODIFIED ARTIFICIAL NEURAL NETWORK SELF ORGANIZING MAP* (*MODIFIED ANNSOM*)

ALBERT SUDARYANTO
07111550020007

DOSEN PEMBIMBING
Prof. Dr. Ir. Achmad Jazidie, M.Eng.
Ir. Rusdhianto Effendie A.K., M.T.

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK SISTEM PENGATURAN
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

TESIS – TE142599

PENGENDALIAN DISTRIBUSI *QUADROTOR SWARM* UNTUK PELACAKAN TITIK *CENTROID* DENGAN MENGGUNAKAN *MODIFIED ARTIFICIAL NEURAL NETWORK SELF ORGANIZING MAP* (*MODIFIED ANNSOM*)

ALBERT SUDARYANTO
07111550020007

DOSEN PEMBIMBING
Prof. Dr. Ir. Achmad Jazidie, M.Eng.
Ir. Rusdhianto Effendie A.K., M.T.

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK SISTEM PENGATURAN
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

LEMBAR PENGESAHAN

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Teknik (M.T.)
di
Institut Teknologi Sepuluh Nopember

oleh:

Albert Sudaryanto
NRP. 07111550020007

Tanggal Ujian : 21 Desember 2017
Periode Wisuda : Maret 2018

Disetujui oleh:

1. Prof. Dr. Ir. Achmad Jazidie, M.Eng. (Pembimbing I)
   NIP: 1959 0219 1986 1010 01

2. Ir. Rusdhianto Effendie A.K., M.T. (Pembimbing II)
   NIP: 1957 0424 1985 0210 01

3. Prof. Dr. Ir. Mohammad Nuh, DEA. (Penguji)
   NIP: 1959 0617 1984 0310 02

4. Dr. Trihastuti Agustinah, S.T., M.T. (Penguji)
   NIP: 1968 0812 1994 0320 01

Dekan Fakultas Teknologi Elektro

Dr. Tri Arief Sardjono, S.T., M.T.
NIP. 1970 0212 1995 1210 01

Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi keseluruhan Tesis saya dengan judul "**PENGENDALIAN DISTRIBUSI *QUADROTOR SWARM* UNTUK PELACAKAN TITIK *CENTROID* DENGAN MENGGUNAKAN *MODIFIED ARTIFICIAL NEURAL NETWORK SELF ORGANIZING MAP* (*MODIFIED ANNSOM*)**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Desember 2017

Albert Sudaryanto

NRP. 07111550020007

Halaman ini sengaja dikosongkan

PENGENDALIAN DISTRIBUSI *QUADROTOR SWARM* UNTUK PELACAKAN TITIK *CENTROID* DENGAN MENGGUNAKAN *MODIFIED ARTIFICIAL NEURAL NETWORK SELF ORGANIZING MAP* (*MODIFIED ANNSOM*)

Nama mahasiswa : Albert Sudaryanto
NRP : 07111550020007
Pembimbing : 1. Prof. Dr. Ir. Achmad Jazidie, M.Eng.
2. Ir. Rusdhianto Effendie A.K., M.T.

ABSTRAK

Akhir-akhir ini, *quadrotor swarm* menjadi topik yang menarik untuk dieksplorasi karena mengarah pada pemanfaatan teknologi nano. Permasalahan utama dalam aplikasi *quadrotor swarm* adalah distribusi antar anggota *quadrotor* dalam suatu kawanan. Salah satu metode yang pernah digunakan adalah *ANN-Self Organizing Map* (*ANNSOM*). Namun, metode tersebut memiliki kelemahan berupa tidak adanya besaran kerapatan antar anggota *quadrotor* dalam area yang dicapai. Di sisi lain, metode *ANNSOM* tergolong efektif untuk diterapkan pada jumlah anggota kawanan *quadrotor* yang banyak.

Pada penelitian ini, metode yang digunakan adalah *ANN-Self Organizing Map* yang telah dimodifikasi, disebut dengan *Modified ANN-Self Organizing Map*. Modifikasi yang dilakukan adalah dalam bentuk pengembangan neuron yang merupakan bagian dari optimasi *feature map* dengan menggunakan metode *Particle Swarm Optimization* (PSO). *ANNSOM* digunakan untuk mengidentifikasi pergerakan awal anggota *quadrotor swarm*. Saat telah teridentifikasi, pergerakan *quadrotor swarm* menuju ke titik *centroid* akan dilakukan dengan menggunakan metode PSO. Untuk menghindari tabrakan antar anggota *quadrotor swarm* pada titik *centroid*, jarak masing-masing anggota *quadrotor* ditentukan menggunakan persamaan jarak *euclidean*.

Hasil penelitian ini menunjukkan bahwa dengan menggunakan metode *Modified ANNSOM*, anggota kawanan *quadrotor* mampu bergerak secara signifikan untuk melacak posisi titik *centroid* secara optimal. Berdasarkan sepuluh data sampel, diketahui bahwa nilai varian untuk koordinat posisi yang dicapai oleh kawanan *quadrotor* dalam proses pelacakan titik *centroid* pada titik *x* adalah 0.0001 dan titik *y* adalah 0.0051 dapat dicapai dengan *time constant delay* 0.5 detik. Selain itu, anggota kawanan *quadrotor* mampu menghindari terjadinya tabrakan dengan anggota yang lain. Saat melakukan proses pelacakan titik *centroid*, nilai rata-rata jarak antar anggota kawanan *quadrotor*, berdasarkan data varian dari sepuluh data sampel, adalah 0.002. Penelitian ini dapat dikembangkan lebih lanjut dengan menambahkan algoritma untuk membentuk formasi yang berbeda-beda dalam dimensi ruang, sehingga dapat diperoleh pola yang paling efektif untuk menampung jumlah *quadrotor* yang banyak.

Kata kunci: *Quadrotor Swarm*, Modified *ANNSOM*, Pelacakan *Centroid*, Distribusi optimal

Halaman ini sengaja dikosongkan

QUADROTORS SWARM DISTRIBUTION CONTROL FOR CENTROID TRACKING USING MODIFIED ARTIFICIAL NEURAL NETWORK SELF ORGANIZING MAP (MODIFIED *ANNSOM*)

By : Albert Sudaryanto
Student Identity Number : 07111550020007
Supervisor(s) : 1. Prof. Dr. Ir. Achmad Jazidie, M.Eng.
2. Ir. Rusdhianto Effendie A.K., M.T.

ABSTRACT

Recently, quadrotor swarm has become an interesting topic to explore because it leads to the utilization of nano technology. The main problem in the application of quadrotor swarm is the distribution between quadrotor members in a swarm. One of the methods ever used was ANN-Self Organizing Map (ANNSOM). However, the method has a weakness in the absence of the magnitude of the density between quadrotor members in a particular area. On the other hand, the ANNSOM method is effective for the large number of quadrotor swarms.

This research used a modified version of ANN-Self Organizing Map, called as Modified ANN-Self Organizing Map. The modifications are in the form of neuron development which is part of feature map optimization using Particle Swarm Optimization (PSO) method. ANNSOM is used to identify the initial movement of quadrotor swarm members. Once identified, the movement of the quadrotor swarm to the centroid will be done using the PSO method. To avoid collisions between members of the swarm quadrotor at the centroid point, the distance of each quadrotor member is determined using the euclidean distance equation.

This study show that by using the ANNSOM Modified method, quadrotor swarm members are able to move significantly to track the position of the centroid optimally. Based on the ten sample data, it is known that the variance value of the coordinates, achieved by the quadrotor swarm in the process of tracking the centroid point at point x is 0.0001 and the y point, is 0.0051 and it can be achieved with a time constant delay of 0.5 seconds. In addition, quadrotor swarm members are able to avoid collisions with other members. When performing the centroid tracking process, the mean value of distance between quadrotor swarm members, based on the variance data of the ten sample data, is 0.002. This research can be further developed by adding algorithms to form different formations in the spatial dimension, so that the most effective pattern can be obtained to accommodate large quantities of quadrotor.

Keywords: Quadrotor Swarm, Modified *ANNSOM*, Centroid Tracking, Optimal Distribution

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur ke hadirat Allah atas kasih sayang dan rahmat-Nya, sehingga Tesis ini dapat terselesaikan dengan baik. Tesis ini disusun untuk memenuhi sebagian persyaratan guna menyelesaikan pendidikan Magister pada Bidang Studi Teknik Sistem Pengaturan, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember (ITS) Surabaya dengan judul:

"**PENGENDALIAN DISTRIBUSI *QUADROTOR SWARM* UNTUK PELACAKAN TITIK *CENTROID* DENGAN MENGGUNAKAN *MODIFIED ARTIFICIAL NEURAL NETWORK SELF ORGANIZING MAP* (*MODIFIED ANNSOM*)**"

Penulis mengucapkan terima kasih kepada semua pihak yang banyak membantu baik secara langsung maupun tidak langsung, sehingga Tesis ini dapat terselesaikan. Terima kasih kepada Prof. Dr. Ir. Achmad Jazidie, M.Eng. dan Ir. Rusdhianto Effendie A.K., M.T. selaku pembimbing Tesis atas segala bimbingan dan motivasi dari beliau hingga terselesaikannya Tesis ini.

Terimakasih kepada Bapak, Ibu, Saudara dan Istri Penulis yang telah memberikan doa dan dukungan. Terimakasih kepada Sulfan Bagus Setyawan dan Hanum Arrosida sebagai teman diskusi dalam proses penyelesaian Tesis. Semoga buku Tesis ini dapat memberikan manfaat bagi pembaca pada umumnya dan mahasiswa Jurusan Teknik Elektro pada khususnya.

Surabaya, 4 Desember 2017

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Quadrotor merupakan salah satu jenis pesawat tanpa awak yang memiliki potensi besar untuk terus dieksplorasi. Banyak hal yang dikaji dalam pengendalian gerak *quadrotor*, yaitu pengendalian kestabilan gerak *quadrotor*, pengendalian *tracking* pada *quadrotor*, dan penerapan *quadrotor* dalam bentuk kawanan. Penelitian tentang *quadrotor* bertujuan untuk menemukan cara yang optimal dalam memanfaatkan *quadrotor* agar dapat digunakan untuk meringankan pekerjaan manusia [1].

Pembahasan mengenai pengembangan *quadrotor* dalam bentuk kawanan menjadi topik yang menarik untuk dieksplorasi karena teknologi *quadrotor* mengarah ke teknologi nano. Sehingga dalam penerapannya diperlukan sebuah metode khusus untuk menangani permasalahan distribusi, koordinasi, formasi, dan komunikasi antar *quadrotor*. Di masa mendatang, diharapkan dapat tercipta suatu teknologi *quadrotor aggregation community* yang menyerupai karakteristik manusia dan hewan dalam aspek kecenderungannya untuk hidup secara berkelompok.

Implementasi *Artificial Neural Network Self-Organizing Map* untuk menghasilkan posisi koordinat yang tepat pada masing-masing anggota *quadrotor swarm*. Pada paper ini menggunakan metode SOM (*Self Organizing Map*) dengan *unsupervised training learning*. Dengan menggunakan metode ini, anggota kawanan dalam jumlah banyak dapat ditampung dengan baik pada berbagai bentuk luasan 3 dimensi. Namun demikian, metode ini juga memiliki beberapa kelemahan, yakni iterasi yang dibutuhkan menjadi lebih besar, ada beberapa anggota *quadrotor swarm* yang keluar dari area yang ditentukan, serta tidak adanya besaran kerapatan antar anggota *quadrotor* dalam area yang dicapai sehingga formasi yang terbentuk menjadi acak [2].

Metode yang lainnya adalah penggunaan algoritma genetika untuk pelacakan titik *centroid* pada kasus *quadrotor swarm.* Penggunaan metode ini

masih kurang optimal, karena algoritma yang diterapkan hanya akan memiliki performa optimal jika anggota *quadrotor swarm* dalam jumlah sedikit dan jumlah generasinya (iterasi) sedikit. Hal ini disebabkan algoritma genetika yang digunakan terjebak pada kondisi lokal optimal yang dianggap merupakan solusi optimalnya, padahal sebenarnya algoritma tersebut belum mencapai kondisi global optimal [3].

Pembahasan selanjutnya yaitu, penggunaan metode *Smoothed Particle Hydrodynamics* (SPH) untuk kasus *quadrotor swarm* dimana *quadrotor* diasumsikan sebagai partikel penyusun zat cair. Sehingga dalam aplikasinya agregasi *quadrotor* menyerupai karakteristik zat cair dalam mengisi suatu wadah dengan batasan berbentuk bola dan kubus. Pengujian secara simulasi pada metode ini menunjukkan hasil yang baik karena anggota kawanan *quadrotor* dengan algoritma SPH dapat mengikuti kontur dari wadah yang ditentukan. Namun dalam pengujian melalui implementasi *swarm* dengan batasan secara fisik menjadikan *quadrotor* sulit untuk segera memenuhi permukaan bawah dari suatu wadah. Faktor lain yang menjadikan *quadrotor* sulit untuk berada di atas *quadrotor* yang lain adalah faktor turbulensi udara dan gaya angkat pada masing-masing baling-baling. Cara mengimplementasikan kawanan quadrotor dalam suatu wadah adalah dengan memberikan penghalang berbentuk silinder pada masing-masing baris sehingga dapat menghindarkan mereka dari tumpukan dan tabrakan [4].

Pada pengembangan kawanan *quadorotor* menggunakan metode-metode tersebut di atas, masing-masing memiliki kelemahan dan kelebihan. Metode *Artificial Neural Network Self-Organizing Map* (*ANNSOM*) memiliki kelemahan berupa tidak adanya besaran kerapatan antar anggota *quadrotor* dalam area yang dicapai sehingga formasi akhir yang terbentuk adalah acak. Namun metode *ANNSOM* tergolong efektif untuk diterapkan pada jumlah anggota kawanan *quadrotor* yang banyak. Sementara, metode algoritma genetika memiliki kelemahan berupa waktu iterasi yang lama, serta belum efektif untuk jumlah anggota *swarm* yang banyak. Namun demikian, algoritma genetika dapat digunakan untuk memprediksi koordinat posisi selanjutnya yang harus dicapai oleh masing-masing anggota *swarm* sehingga dapat mengikuti pergeseran posisi titik *centroid.*

Oleh karena itu, akan diusulkan Tesis yang meneliti tentang pengendalian distribusi *quadrotor* untuk pelacakan titik *centroid* pada kawanan *quadrotor*

menggunakan metode *artificial neural network self organizing map* yang telah dimodifikasi (*modified ANNSOM*). Metode ini menangani aspek pengaturan distribusi kawanan *quadrotor* dengan cara melakukan perhitungan jarak terhadap titik *centroid* agar dapat melacak posisi *centroid* secara optimal dan dapat menghindari tabrakan antar anggota kawanan *quadrotor.*

1.2 Rumusan Masalah

Pada aspek distribusi kawanan *quadrotor* terdapat dua hal penting yang harus diselesaikan. Pertama, terkait dengan kemampuan setiap anggota kawanan untuk memposisikan diri di sekitar titik *centroid* yang ditentukan. Kedua, terkait dengan kemampuan setiap anggota kawanan dalam menghindari terjadinya tabrakan antar antar anggota kawanan *quadrotor*. Dengan mempertimbangkan kedua permasalahan tersebut, maka yang menjadi rumusan masalah dalam Tesis ini adalah bagaimana memodifikasi metode *ANNSOM* untuk pengendalian distribusi kawanan *quadrotor* agar dapat melacak posisi titik *centroid* secara optimal dapat dicapai dengan *time constant delay* minimum yaitu berada pada posisi terdekat dengan titik *centroid* dan dapat menghindari tabrakan antar anggota kawanan *quadrotor* saat proses pelacakan titik *centroid*.

1.3 Tujuan

Penelitian ini bertujuan untuk menghasilkan desain sistem kawanan *quadrotor* yang dapat melacak posisi titik *centroid* dengan penyebaran yang optimal dan dapat menghindari tabrakan antar anggota kawanan *quadrotor*. Serta memperhatikan *time constant delay* dari metode kontrol yang diterapkan. Sehingga kondisi ideal dari distribusi kawanan *quadrotor* dalam melacak posisi titik *centroid* dapat dicapai dengan *time constant delay* minimum.

1.4 Batasan Masalah

Peneilitian ini fokus pada metode distribusi *swarm* pada *quadrotor* dalam melacak posisi titik *centroid* secara optimal, sehingga ditentukan batasan masalah sebagai berikut,

1. Tidak memperhitungkan energi yang digunakan dalam pergerakan *quadrotor*.
2. Dengan asumsi bahwa pergerakan *quadrotor* pada sumbu *z* dengan ketinggian yang tetap, sehingga pergerakan *quadrotor swarm* direpresentasikan pada bidang koordinat *x* dan *y*.
3. Tidak membahas nilai kecepatan masing-masing anggota *quadrotor swarm* saat melakukan pembelokan arah untuk menghindari terjadinya tabrakan.

1.5 Kontribusi

Penelitian ini dilakukan untuk mendapatkan metode yang tepat untuk mengatasi permasalahan distribusi *quadrotor* dalam sebuah kawanan dengan menggunakan metode *ANNSOM* yang telah dimodifikasi. Adapun modifikasi yang dilakukan adalah dengan menambahkan fitur *centroid tracking accuracy* yang memperhatikan *time constant delay* dari metode kontrol yang diterapkan. Sehingga distribusi kawanan *quadrotor* dalam upaya melacak posisi titik *centroid* dapat dicapai dengan *time constant delay* yang minimum.

BAB 2
KAJIAN PUSTAKA

Bab ini berisi hasil kajian pustaka terdahulu yang berkaitan dengan permasalahan distribusi *quadrotor swarm*, yakni meliputi dua metode kontrol yang telah diaplikasikan pada penelitian-penelitian sebelumnya, yaitu *Artificial Neural Network Organizing Map* (*ANNSOM*) dan *Genetic Alogarithm*. Pada bab ini, juga dibahas tentang dasar teori yang digunakan sebagai referensi pendukung dalam menyelesaikan Tesis.

2.1 Kajian Penelitian Terkait

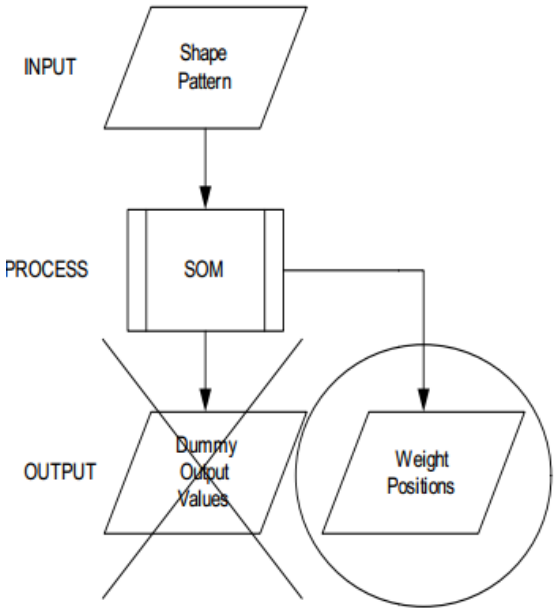
Yang termasuk dalam penelitian terkait dalam penelitian ini adalah berupa kajian-kajian yang berhubungan dengan pengendalian sistem *centroid* pada kawanan *quadrotor* beserta kajian penelitian lainnya yang terkait dengan kawanan *quadrotor*.

2.1.1 Obstacle Avoidance for Quadrotor Swarm using Artificial Neural Network Self-Organizing Map [2]

Pada paper [2] penulis merancang *Artificial Neural Network Organizing Map* untuk menghasilkan koordinat lokasi yang tepat pada masing-masing anggota *quadrotor swarm* sehingga apabila diberikan area luasan tertentu maka *quadrotor swarm* akan mengikuti koordinat tersebut tanpa bertabrakan. Pada pengujian terdapat tiga bentuk luasan dimana semua *quadrotor swarm* harus berada di dalam luasan tersebut.

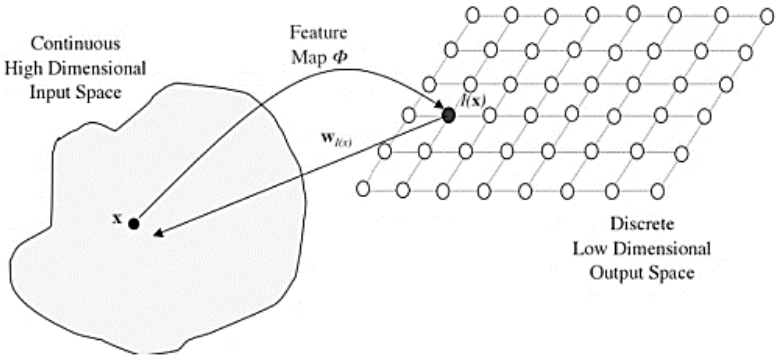
Pada umumnya ANN membutuhkan *rigorius training* atau pembelajaran supervisori agar berkerja dengan optimal, tetapi pada paper ini menggunakan *unsupervised training learning* melalui metode SOM (*Self Organizing Map*) dimana proses *training* sangat terbatas. Metode ini bekerja dengan cara mentransformasi dimensi sistim masukan kontinyu ke dalam sistim diskrit, yang nantinya dari domain diskrit dapat dikembangkan *feature* mapnya. Sehingga untuk mengembangkan bentuk *feature map* diperlukan *competitive learning model* diantara neuron.

Sistem yang digunakan pada paper [2] menggunakan sistem IPO (*Input-Proses-Output*). Pemindaian bentuk *map* tidak di bahas pada paper ini. Bobot dari neuron adalah keluaran aktual yang harus dirancang agar anggota kawanan *quadrotor* dapat memposisikan dirinya dengan tepat sesuai koordinat yang diberikan.



Gambar 2. 1 Bentuk Model Sistem IPO [2]

Gambar 2.2 menjelasakan proses pembentukan dari *feature map* dari masukan dimensi kontinyu menuju keluaran dimensi diskrit.

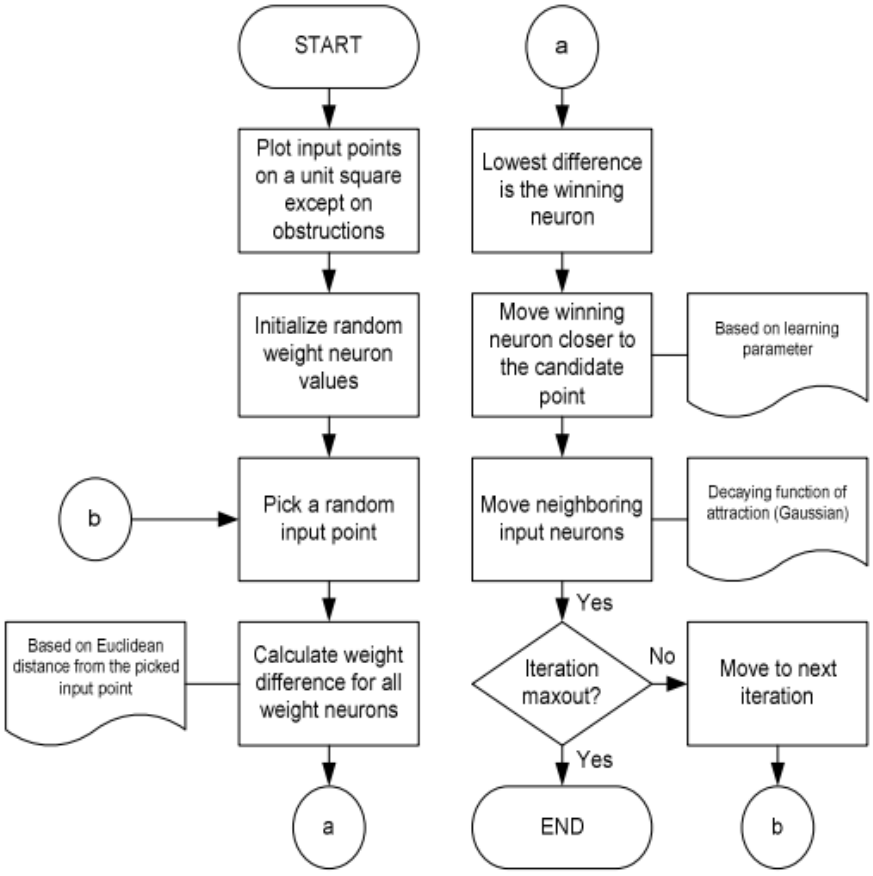


Gambar 2. 2 *Input-Output Space Feature Map* [2]

Pada penelitian ini, bentuk dari bidang kerja telah ditentukan sebelumnya yaitu menggunakan bentuk kubus dengan batas-batas yang telah di tentukan. Halangan dinyatakan dalam tiga bentuk unik sebagai masukan vektor yang digambarkan secara acak di dalam bidang kubus.

Anggota kawanan *quadrotor* di inisialiasasi secara acak dengan nilai bobot $\overline{w}$. Bobot tersebut berisi koordinat 3D pada masing-masing neuron, nilai *vector* bobot $\overline{w}$ harus mempunyai jarak yang sesuai dari bentuk masukan *vector* dan harus berada di dalam luasan unit kubus. Dalam proses kompetisi, salah satu *vector* acak yang terpilih dijadikan sebagai vektor $\overline{pick}$. Jarak Euclidean antara bobot $\overline{w}$ dan $\overline{pick}$ dikalkulasi pada masing-masing neuron dengan menggunakan Persamaan 2.1.

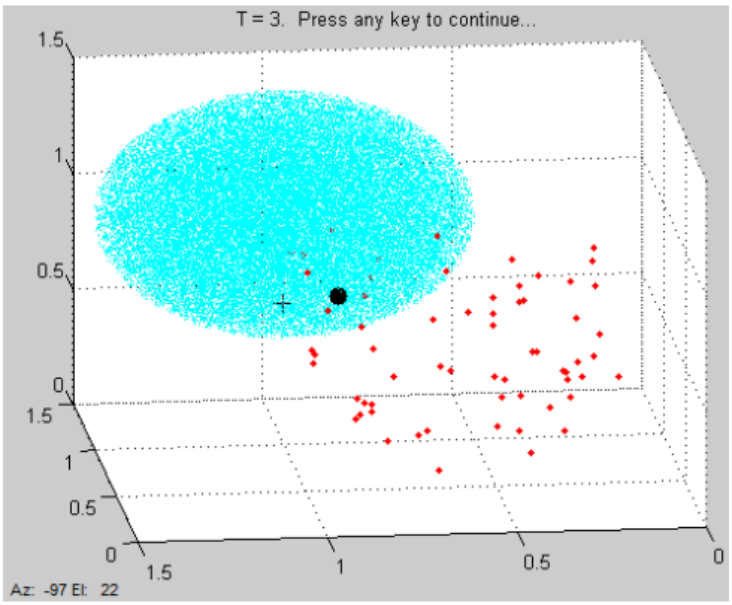
$$w_{diff} = \left|\overline{w} - \overline{pick}\right| \tag{2.1}$$



Gambar 2. 3 Flowchart dari Metode *Self Organizing Map* [2]

Neuron yang mempunyai nilai terkecil dari w_{diff} merupakan pemenang neuron. Proses pembelajaran dalam kompetisi ini merupakan transformasi dari kontinyu ke diskrit. Fungsi *neighbourhood* menggunakan fungsi distribusi gaussian. Diagram alur keseluruhan metode SOM dapat dilihat pada Gambar 2.3.

Pada paper ini, pengujian dilakukan dengan menggunakan tiga bentuk pola berbeda dengan jumlah *grid* yang berbeda pula. Pada gambar 2.4 merupakan contoh iterasi menggunakan *feature map grid* 4x4x4 nilai $lr_f = 0.1$ $lr_i = 0.5$ $\sum_i = 3$ $\sum_f = 0.1$ dan iterasi maksimum adalah 2000 iterasi, serta menggunakan bobot $\overline{w}$ (0.5, 0.5, 0.5).

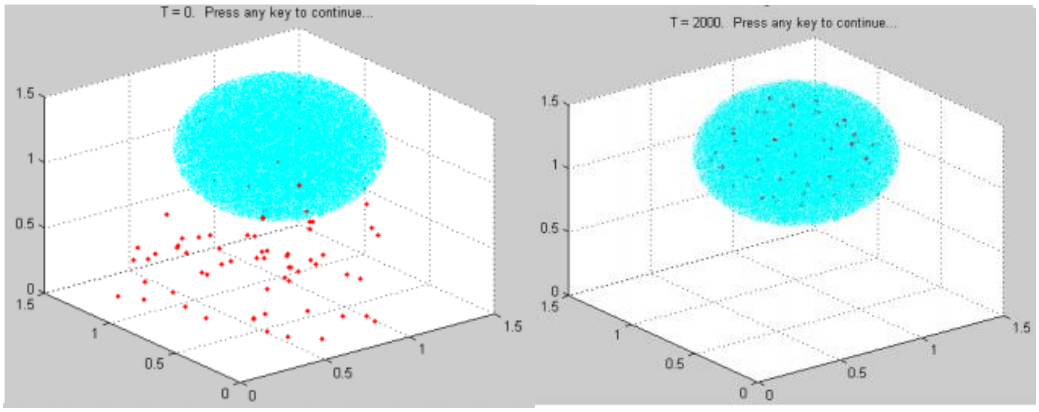


Gambar 2. 4 Contoh Iterasi Sesuai dengan Simulasi pada Paper [2]

Titik merah merupakan bobot neuron yang dilakukan penggambaran secara acak pada unit kubus dengan koordinat (0.5, 0.5, 0.5). Titik bundar biru merupakan masukan vektor yang juga di gambar secara acak dengan koordinat titik pusat di (1.0, 1.0, 1.0) pada unit kubus. Tanda silang merupakan lokasi pengambilan masukan vektor untuk iterasi. Neuron terpilih akan di tandai dengan titik hitam.

Pengujian selanjutnya dilakukan dengan pola berbentuk bola. Tujuan dari pengujian ini agar *quadrotor swarm* dapat tergambar secara acak di titik (0.5, 0.5,

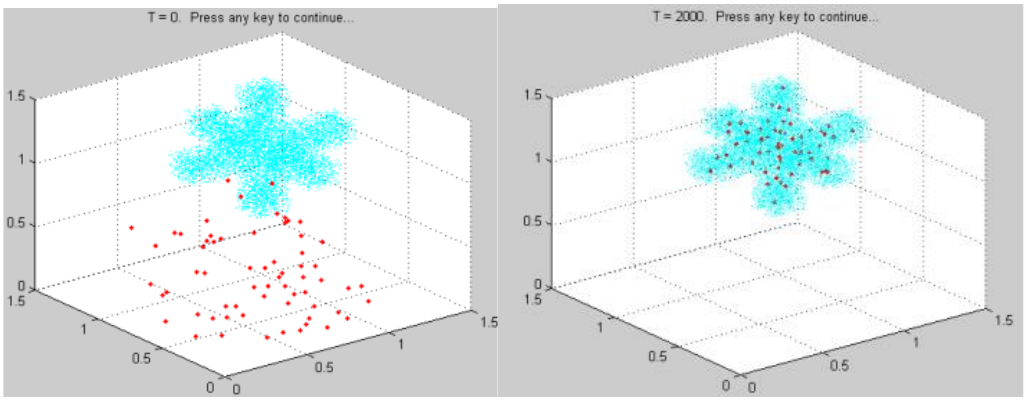
0.5) pada unit kubus, dan bergerak menuju posisi (1.0, 1.0, 1.0) pada pola yang berbentuk bola.



Gambar 2. 5 Sebelum dan Setelah Implementasi Pola Berbentuk Bola

Pada Gambar 2.5 menjelasakan pengaturan sebelum dan sesudah algoritma diimplementasikan. Nilai inisial awal dan akhir dinyatakan dengan sigma 3 dan 0.1. Jumlah iterasi yang digunakan adalah 2000. Berdasarkan informasi yang diperoleh pada Gambar 2.5, diketahui bahwa semua anggota *quadrotor swarm* dapat mengikuti pola yang diberikan sesuai koordinat yang ditentukan.

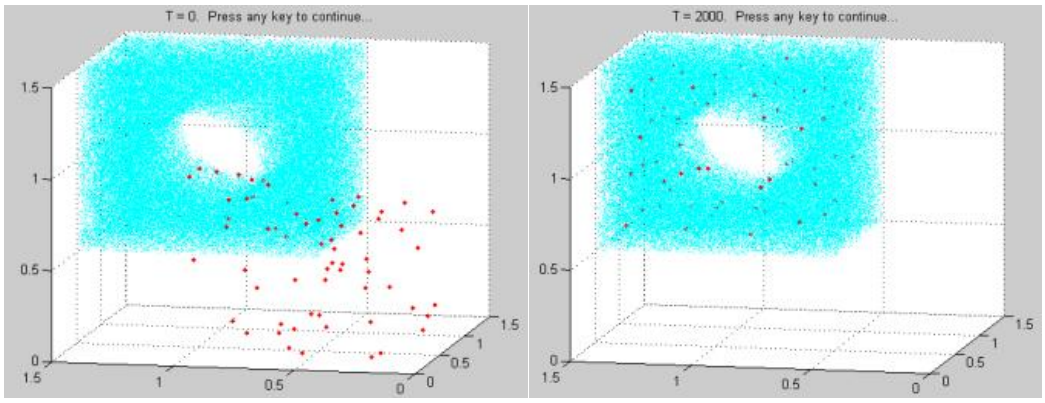
Pengujian kedua menggunakan pola berbentuk silang dimana pola ini juga di pusatkan pada bidang kubus (1.0, 1.0, 1.0), parameter yang digunakan sama dengan pengujian pertama.



Gambar 2. 6 Sebelum dan Setelah Implementasi Pola Berbentuk Silang

Pada hasil pengujian kedua, diketahui bahwa *quadrotor swarm* dapat mengikuti pola masukan vektor yang diberikan. Pengujian ketiga adalah

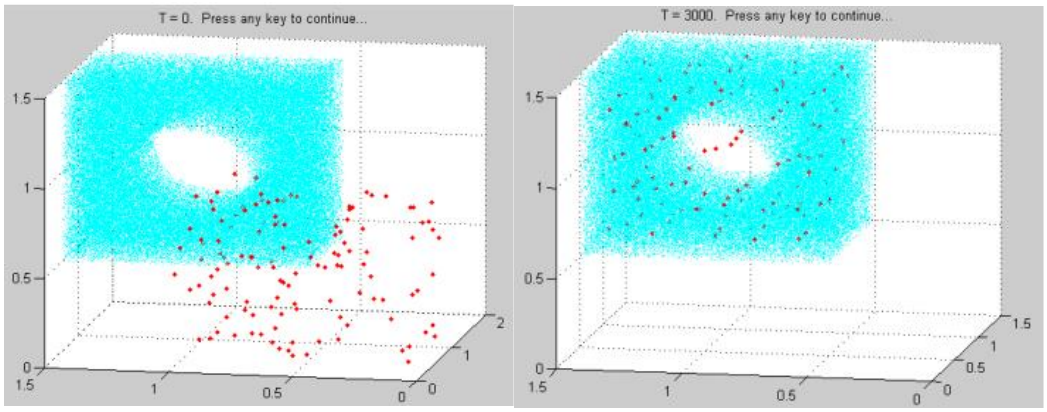
menggunakan pola masukan berbentuk pipa, Gambar 2.7 merupakan hasil pengujian ketiga.



Gambar 2. 7 Sebelum dan Setelah Implementasi Pola Berbentuk Pipa

Hasil pada gambar pengujian ketiga juga mirip dengan pengujian pertama dan kedua dimana semua anggota *swarm* dapat masuk mengikuti pola yang diberikan.

Permasalahan yang di temukan dalam metode SOM ini adalah ketika jumlah *grid* yang masih sedikit dapat mengikuti pola, dan apabila jumlah *grid* terus dinaikan maka terdapat beberapa *quadrotor* yang tidak memenuhi ketentuan dimana *quadotor* berada di luar pola yang di berikan. Pada pengujian ini diberikan jumlah *grid* sebanyak 5 dan iterasi yang di gunakan sebesar 3000 dengan menggunakan bentuk pola pipa.



Gambar 2. 8 Pengujian Metode SOM untuk 5 *Grid* dan 3000 Iterasi

Berdasarkan Gambar 2.8 diketahui bahwa, beberapa anggota *quadrotor swarm* keluar dari pola masukan vektor yang diberikan. SOM memiliki kelemahan pada iterasi yang dibutuhkan cukup besar dan terdapat beberapa anggota *quadrotor swarm* yang keluar dari area yang ditentukan. Serta tidak mempertimbangkan kerapatan antar anggota *quadrotor* dalam area yang dicapai sehingga formasi yang terbentuk adalah acak.

2.1.2 A genetic Algorithm Approach to Swarm Centroid Tracking in Quadrotor Unmanned Aerial Vehicles [3]

Agregasi atau suatu kesatuan dalam kelompok merupakan hal yang utama dalam suatu persoalan *swarm*. Hal yang terpenting dalam agregasi adalah kemampuan setiap anggota kawanan untuk berada di sekitar titik spesifik yang ditentukan. Tujuan dari penerapan algoritma genetika adalah menjaga objek agar tetap melintas pada titik pusat *swarm*, saat dua parameter diminimisasi yaitu jarak pada setiap *quadrotor* dan jarak masing-masing *quadrotor* terhadap objek.

Secara alami, laron bergerak di sekitar sumber cahaya dan lebah selalu berkoloni untuk melindungi sarangnya dengan cara berkerumun di sekitar sarang tersebut. Perilaku alamiah tersebut diadaptasi pada *quadrotor swarm* dengan tujuan menjaga objek agar tetap berada di tengah dengan cara bergerak ataupun diam di sekitar objek tersebut. Karakteristik dari suatu kawanan didefinisikan sebagai kumpulan perilaku pada kelompok individu dengan spesies yang sama untuk mencapai tujuan bersama dengan cara membentuk suatu kesatuan. Dalam suatu algoritma berkoloni, tidak ada pemimpin dan setiap anggota *swarm* bergerak secara mandiri berdasarkan tujuan dan posisi yang bersesuaian dengan anggota yang lain dalam koloni tersebut. Pada paper ini, *swarm* yang disimulasikan akan membentuk satu kesatuan dan menjaga objek tetap berada di tengah-tengah dari pergerakan *swarm* tersebut. Setiap anggota dari *swarm* tersebut akan bergerak dan mengatur posisi mereka berdasarkan pergerakan dari objek.

Algoritma genetika mencari algoritma berdasarkan konsep evolusi secara alami, dan kemampuan bertahan dari individu yang memiliki tingkat *fitness* tinggi. Penentuan koordinat posisi dari *quadrotor* tidak dilakukan secara acak, algoritma ini akan memilih solusi terbaik dari populasi yang awalnya acak dan individu yang

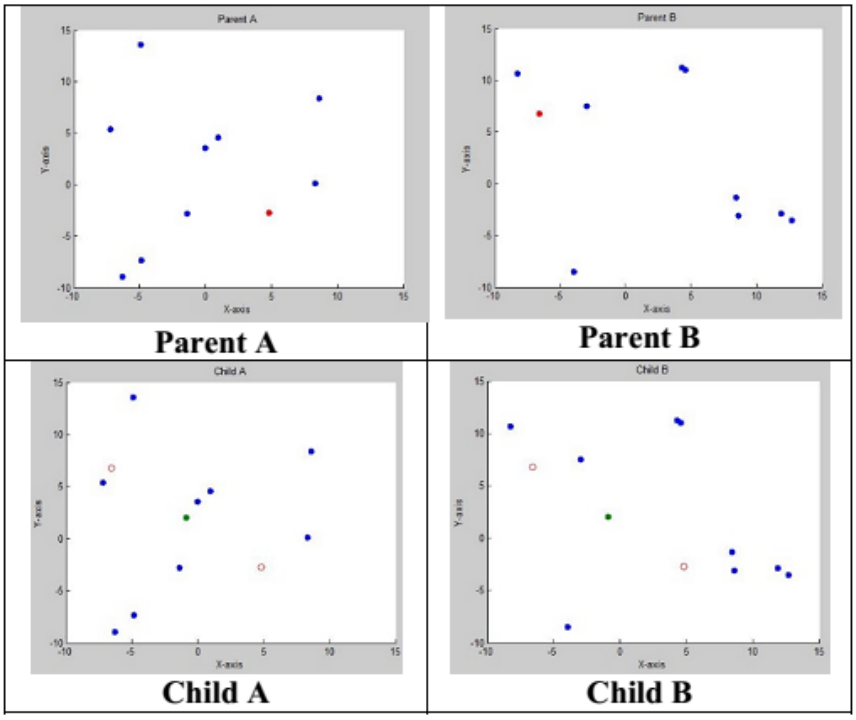
terus berkembang akan dinyatakan sebagai solusi terbaik berdasarkan tingkat *fitness* nya. Metode ini akan menetukan beberapa arah pencarian yang berada di beberapa titik, sehingga mampu menyebar pada posisi yang sesuai agar diperoleh solusi pemecahan masalah yang terbaik pada kasus *swarm*. Langkah kerja dari algoritma genetika yang diterapkan pada pengaturan *swarm* yaitu, ada populasi awal yang diinisialisasikan sebagai generasi pertama pada algoritma. Setiap solusi ditentukan titik koordinat yang menyatakan posisi selanjutnya dari keseluruhan *quadrotor* dalam anggota *swarm*.

Desain sistem pada penelitian ini mencakup beberapa tahapan, yaitu inisialisasi posisi sebagai *input* dan posisi selanjutnya merupakam *output*. Algoritma dirancang untuk diterapkan secara nyata, dimana posisi yang dibaca oleh sensor secara akurat merupakan inisialisasi posisi yang berperan sebagai *input*, setelah itu algoritma yang diterapkan akan mencari posisi akhir dari *quadrotor* dan dinyatakan sebagai *output*. Jumlah *quadrotor* yang digunakan dalam *swarm* adalah 10 sampai dengan 100. Pada sistim koordinat cartesian ditentukan koordinat posisi masing masing *quadrotor* dan diberikan batas toleransi pergerakan agar tidak terjadi tabrakan antar *quadrotor*. Inisialisasi parameter yang digunakan adalah koordinat dari setiap *quadrotor* dan koordinat objek *centroid* yang diikuti oleh *quadrotor swarm*. *Output* dari program yang diterapkan adalah koordinat selanjutnya dari setiap *quadrotor* untuk mengikuti pergeseran *centroid*. Tahapan penerapan algoritma genetika pada sistem yaitu,

1) Reproduksi

Reproduksi menggambarkan posisi awal sebagai inisialisasi pertama (*parent*) dan *children* menyatakan posisi selanjutnya. Cara menentukan posisi selanjutnya (*children*) adalah mengambil titik tengah dari posisi awal (*parent*) yang telah di-*mirror* sebelumnya, arah *mirror* mendekati titik *centroid* nya (0,0). Titik merah bulatan penuh pada *parent* menyatakan posisi awal, sedangkan bulatan garis merah kosong pada *child* adalah duplikasi posisi awal untuk menentukan posisi selanjutnya. Titik hijau menyatakan posisi selanjutnya dari *quadrotor* ditentukan dengan cara mengambil titik tengah dari duplikasi posisi awal. Gambar 2.9 menyatakan proses reproduksi pada algoritma genetika yang dinyatakan sebagai

tahapan proses penentuan titik koordinat selanjutnya yang harus dicapai oleh *quadrotor* untuk mengikuti pergerakan objek *centroid*.



Gambar 2. 9 Tahap Penentuan Koordinat Selanjutnya yang Harus Dicapai oleh *Quadrotor* Menggunakan Algoritma Genetika.

2) Fungsi *Fitness*

Fungsi *fitness* digunakan untuk mencari solusi terbaik dalam menentukan koordinat *quadrotor* selanjutnya berdasarkan pergerakan *centroid*. Jarak *centroid* terhadap posisi *quadrotor swarm* dinyatakan dalam Persamaan 2.2

$$d = \sqrt{(X - x_0)^2 + (Y - y_0)^2 + (Z - z_0)^2} \quad (2.2)$$

dengan,

x_0, y_0, dan z_0 menyatakan posisi awal pada sumbu x, y dan z

X, Y, dan Z menyatakan posisi *centroid*

Fungsi *fitness* dirumuskan

$$F = \alpha d + \beta\Delta + \chi\delta \quad (2.3)$$

dengan α, β, χ merupakan konstanta yang bersesuaian dengan d, χ, δ. Nilai konstanta yang ditentukan yaitu $\alpha = 15, \beta = 1, \chi = 1$. Pada kasus ini, nilai fungsi

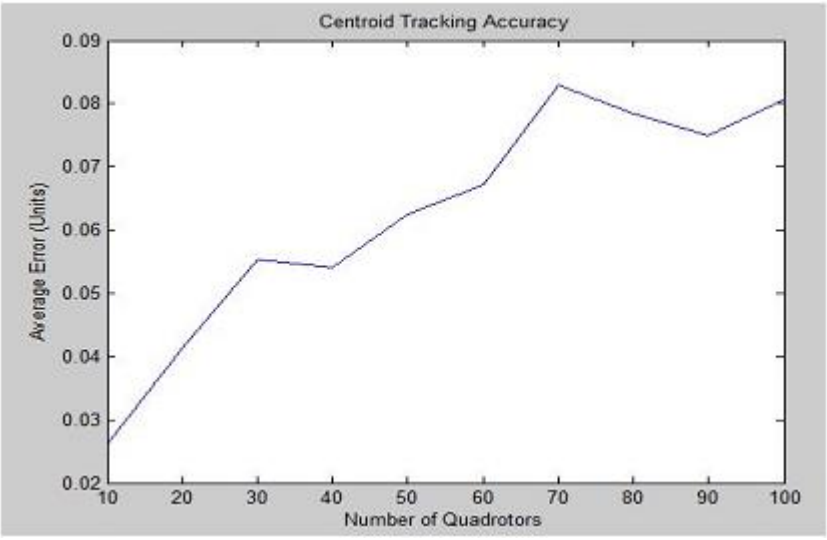
fitness (F) semakin mendekati nilai "1" maka dinyatakan sebagai solusi terbaik dari algoritma yang digunakan.

3) Hasil Pengujian *Tracking Centroid*

Hasil pengujian *tracking centroid* dengan nilai parameter $\alpha = 15, \beta = 1, \chi = 1$ ditunjukkan dalam Tabel 2.1 dan grafik pada Gambar 2.10

Tabel 2. 1 Hasil Tracking *Centroid*

Jumlah *Quadrotor*	Error rata-rata dari 100 iterasi *Generic Alogarithym* (Unit)
10	0.026173
20	0.041306
30	0.055336
40	0.054169
50	0.062468
60	0.067190
70	0.082854
80	0.078457
90	0.074995
100	0.080620



Gambar 2. 10 *Centroid Tracking Accuracy*

Dari hasil pengujian ini dapat diketahui bahwa semakin banyak jumlah anggota *quadrotor swarm*, nilai rata rata error yang dihasilkan semakin besar. Hasil

terbaik diperoleh jika jumlah *quadrotor* maksimal 30, nilai rata-rata error maksimal adalah 0,6.

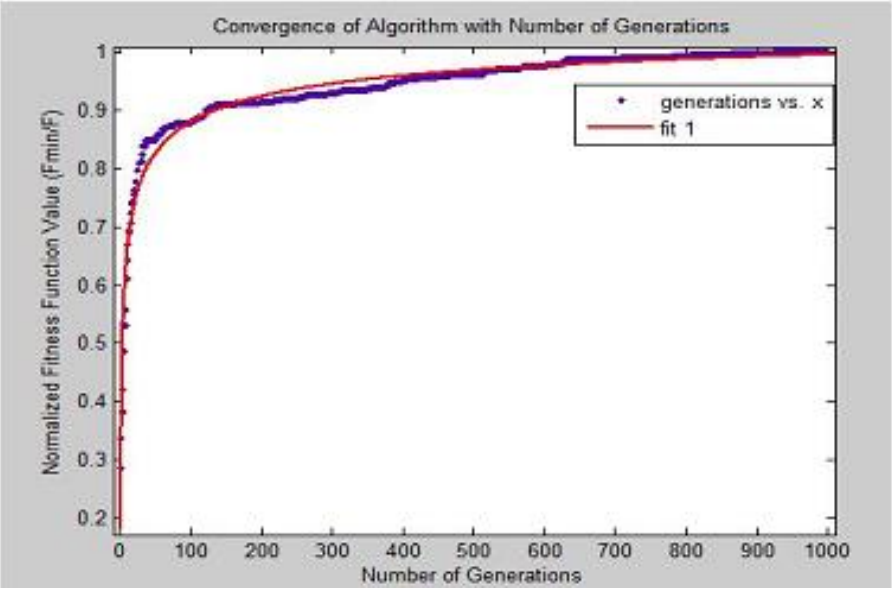
4) Pengujian Jumlah Generasi yang Digunakan

Pengujian dilakukan dengan jumlah generasi sebanyak 1000, dari hasil ini diketahui bahwa nilai *fitness* generasi selanjutnya merupakan hasil normalisasi nilai *fitness* generasi sebelumnya. Nilai *fitness* dinyatakan sebagai solusi terbaik apabila nilai nya mendekati "1" . Dari hasil pengujian ini diketahui bahwa semakin banyak jumlah genarasi maka semakin bagus nilai *fitness* yang dihasilkan. Normalisasi nilai *fitness* dinyatakan dalam rumus Persamaan 2.4

$$F = -0.929x^{-0.3028} + 1.11 \qquad (2.4)$$

dengan F adalah normalisasi nilai fungsi *fitness* dan x adalah jumlah generasi.

Persamaan grafik normalisasi nilai *fitness* terhadap jumlah generasi seperti pada Gambar 2.11,

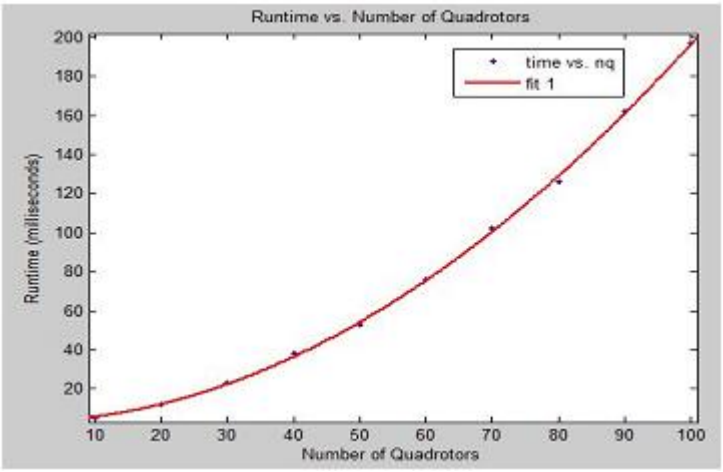


Gambar 2. 11 Grafik Pengujian Nilai Konvergensi Terhadap Jumlah Generasi

Parameter yang menentukan lama waktu eksekusi program adalah jumlah *quadrotor* dan jumlah generasi (iterasi). Semakin banyak jumlah *quadrotor swarm*, maka waktu yang diperlukan untuk eksekusi juga semakin besar. Data ditunjukkan pada Tabel 2.2 dan grafik pada Gambar 2.12.

Tabel 2. 2 Hasil Runtime Program untuk 30 Generasi

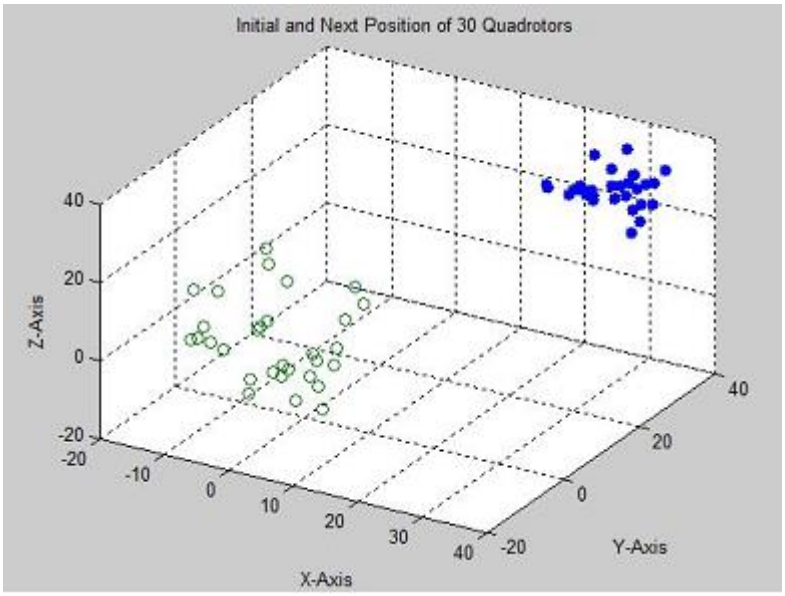
Jumlah *Quadrotor*	*Runtime* (milisekon)
10	5
20	12
30	23
40	38
50	5
60	76
70	102
80	126
90	162
100	197



Gambar 2. 12 Hasil Runtime Program Berdasarkan Jumlah *Quadrotor*

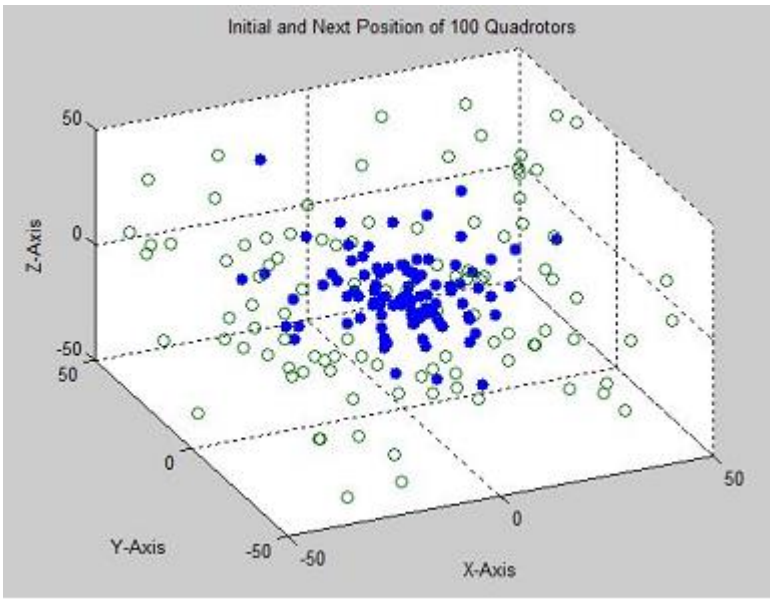
Hasil pengujian menunjukkan bahwa semakin banyak jumlah generasi (iterasi) waktu yang diperlukan untuk eksekusi program juga semakin lama.

Pengujian sistem dilakukan sebanyak empat kali dengan posisi koordinat *centroid* yang berbeda, pengujian pertama untuk koordinat *centroid* pada titik (30,30,30) diperoleh posisi koordinat yang dihitung oleh algoritma genetika (30.007107, 29.951099, 30.014622). Jumlah generasi diatur pada nilai 30 dan jumlah *quadrotor* yang digunakan 30 sehingga waktu eksekusi yang diperlukan 24ms. Gambar 2.13 menunjukkan grafik tiga dimensi pergerakan *swarm* dalam mengitari *centroid.*



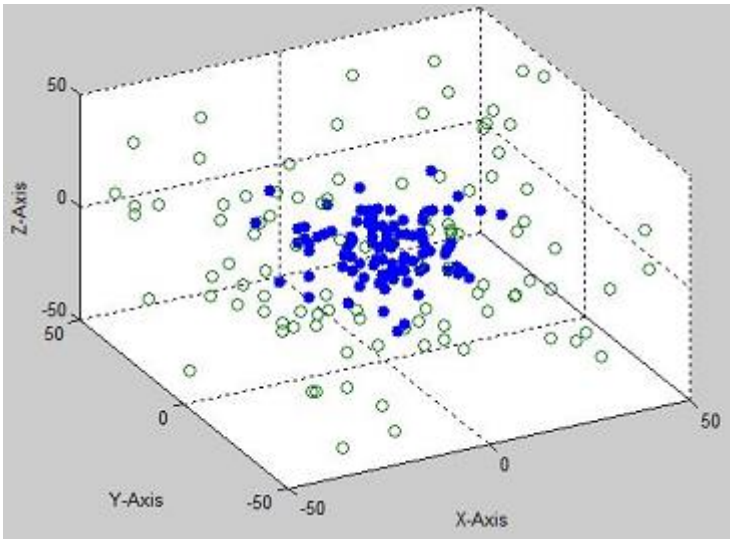
Gambar 2. 13 Pengujian untuk Koordinat *Centroid* (30,30,30)

Pengujian kedua yaitu koordinat *centroid* (0,0,0) Jumlah *quadrotor* yang digunakan 100, jumlah generasi 30, waktu yang diperlukan untuk eksekusi 204 ms. Koordinat *centroid* yang dikalkulasi oleh algoritma genetika adalah (0.001812, 0.035475, -0.058425) data trayektori *swarm* dalam mengikuti *centroid*, dapat dilihat pada grafik Gambar 2.14.



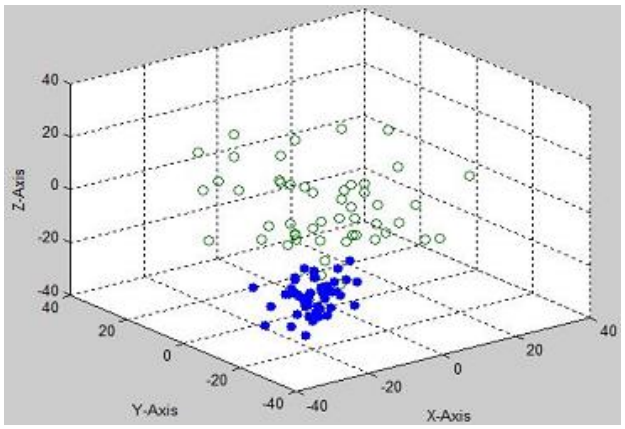
Gambar 2. 14 Pengujian untuk Koordinat *Centroid* (0,0,0) dengan Jumlah Generasi 30

Pengujian ketiga yaitu koordinat *centroid* (0,0,0) Jumlah *quadrotor* yang digunakan 100, jumlah generasi 100, waktu yang diperlukan untuk eksekusi 656 ms. Koordinat *centroid* yang dikalkulasi oleh algoritma genetika adalah (0.001159, -0.000826, -0.001263) data trayektori *swarm* dalam mengikuti *centroid*, dapat dilihat pada grafik Gambar 2.15.



Gambar 2. 15 Pengujian untuk Koordinat *Centroid* (0,0,0) dengan Jumlah Generasi 100

Pengujian terakhir yaitu koordinat *centroid* (-20,-20,-20) Jumlah *quadrotor* yang digunakan 50, jumlah generasi 30, waktu yang diperlukan untuk eksekusi 55 ms. Koordinat *centroid* yang dikalkulasi oleh algoritma genetika adalah (-19.929068, -20.005068, -19.896784) data trayektori *swarm* dalam mengikuti *centroid*, dapat dilihat pada grafik Gambar 2.16.



Gambar 2. 16 Pengujian untuk Koordinat *Centroid* (-20,-20,-20)

Pada paper ini dilakukan pengujian karakteristik algoritma *swarm* menggunakan algoritma genetika untuk menetukan agregasi dan algoritma *centroid tracking*. Efektifitas program yang diterapkan dievaluasi berdasarkan akurasi *centroid tracking* dan waktu eksekusi program. Akurasi *centroid tracking* memiliki rata–rata error 0.0623568 untuk pengujian jumlah anggota *quadrotor swarm* 10 hingga 100. Hasil ini diperoleh dengan nilai pembobotan α sebesar 15. Akurasi *centroid tracking* bergantung pada nilai bobot α. Semakin besar nilai α, maka hasil tracking *centroid* semakin presisi. Hasil pengujian simulasi juga menunjukkan bahwa nilai rata-rata error akan meningkat, jika jumlah *quadrotor* yang digunakan dalam *swarm* semakin banyak. Pengujian waktu eksekusi program dan tingkat akurasi tracking *centroid* dengan menggunakan algoritma ini sesuai untuk penerapan jumlah *swarm* yang sedikit.

Penggunaan algoritma genetika untuk *swarm* algorithm masih kurang optimal, karena algoritma yang diterapkan dapat memiliki performa optimal, jika jumlah anggota *quadrotor swarm* sedikit dan generasinya (iterasi) sedikit. Hal ini bisa disebabkan karena algoritma genetika yang digunakan terjebak pada kondisi *local optimal* yang dianggap merupakan solusi optimalnya, padahal sebenarnya algoritma tersebut belum mencapai kondisi *global optimal*.

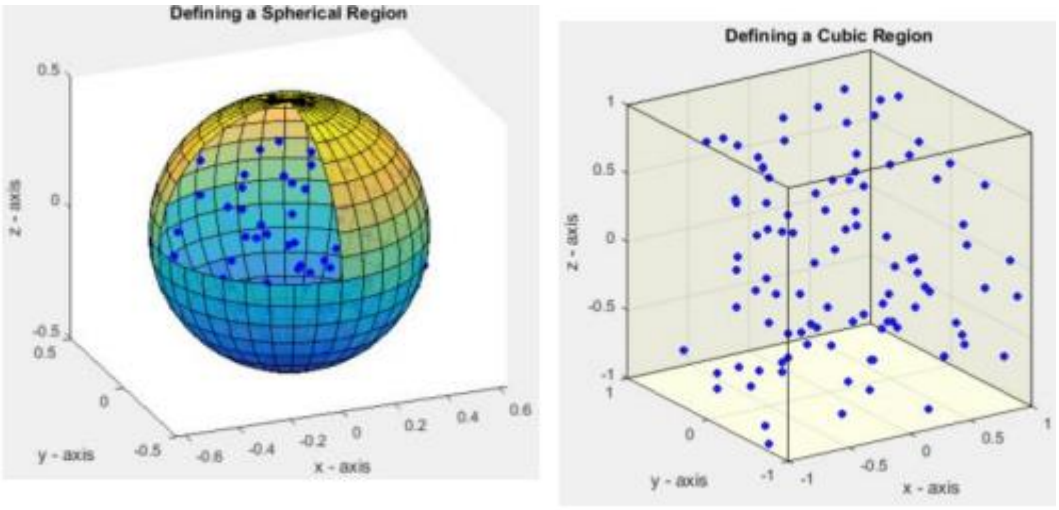
2.1.3 Implementation of varied particle container for Smoothed Particle Hydrodynamics-based aggregation for unmanned aerial vehicle *quadrotor* [4]

Karakteristik *swarm* pada umumnya berdasarkan karakteristik makhluk hidup. Sangat jarang penelitian yang membahas *swarm* berdasarkan karateristik benda mati seperti pergerakan partikel dalam suatu ruang tertentu. Paper ini membahas tentang perilaku *swarm* dengan pendekatan mekanika cairan dalam suatu wadah.

Metode *Smoothed Particle Hydrodynamics* (SPH) pertamakali digunakan untuk menyelesaikan permasalahan permodelan *asthropysic*. Pada tahun selanjutnya, SPH digunakan sebagai cara penyelesaian laju aliran suatu cairan agar bebas dari hambatan dengan pendekatan perhitungan matematis. Pada paper ini,

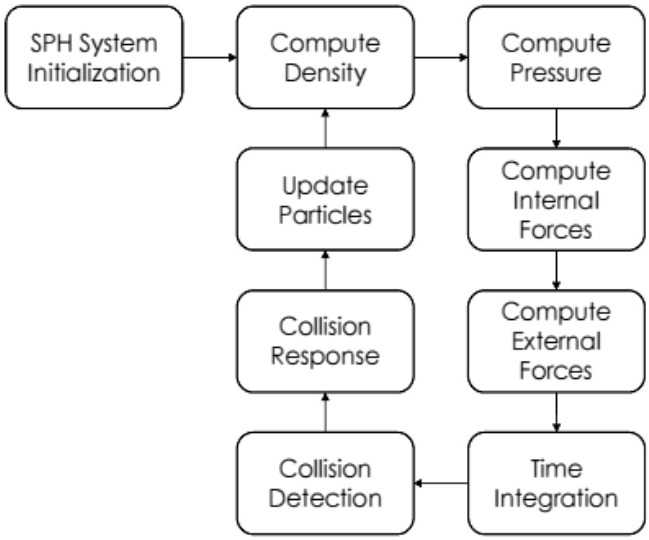
menyajikan aplikasi dari SPH untuk pendekatan metode *swarm* yang diasumsikan sebagai partikel cairan dalam suatu wadah tertentu untuk perilaku agregasi.

Pada studi kasus ini, partikel yang direpresentasikan dalam SPH adalah sistem *quadrotor swarm*. Dimana cairan dalam suatu wadah dinyatakan sebagai *quadrotor swarm*. Partikel SPH dipengaruhi oleh gaya dari luar, gaya gravitasi dan batasan wadah yang digunakan. Wadah/kontainer yang digunakan dalam kasus ini adalah bola dan kubus. Bentuk *swarm* yang diharapkan adalah *quadrotor* dapat mengisi wadah yang ditentukan. Gambar 2.17 menyatakan wadah berbentuk bola dan kubus yang digunakan sebagai wadah penampung *quadrotor* sebagai partikel SPH . Ketika partikel mencapai batasan dari wadah, kecepatan dengan arah berlawanan akan menjadikan partikel didalamnya bergeser sehingga bisa memenuhi wadah tersebut.



Gambar 2. 17 *Container* Berbentuk Bola dan Kubus

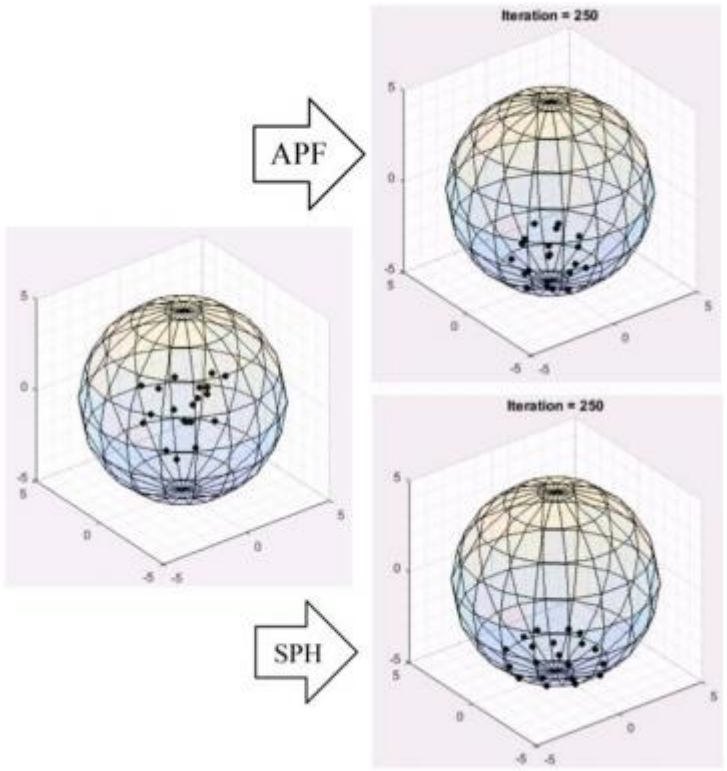
Pada algoritma SPH, keseluruhan proses akan berpengaruh pada perhitungan secara numerik dan penambahan partikel, penambahan waktu yang singkat untuk pembaharuan pergerakan formasi, kenaikan waktu pada masing-masing anggota koloni akan berpengaruh pada tingkat penyebaran yang lambat. Gambar 2.18 merupakan diagram alir algoritma SPH.



Gambar 2. 18 Diagram Alir Algoritma SPH

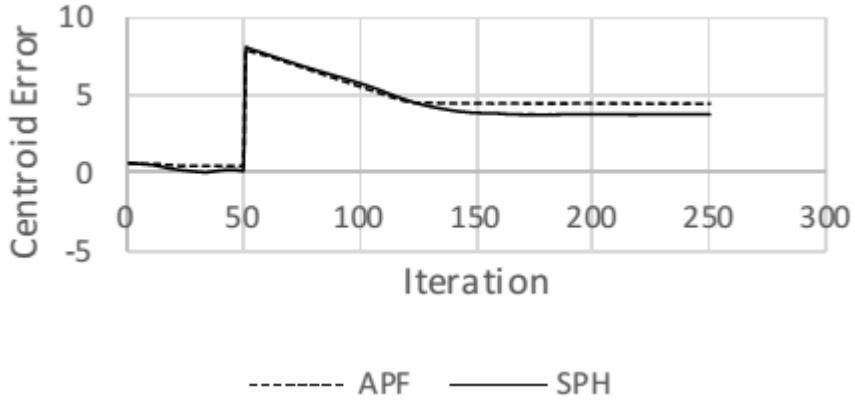
Validasi dari algoritma SPH yang digunakan dengan cara membandingkan dengan algoritma *benchmark*. Selain itu. Akan digunakan *Artificial Potential Function* (APF) sebagai pembanding dasar dengan dimensi 3D ukuran 10x10x10.

Pengujian pertama menggunakan wadah berbentuk bola, titik pusat wadah (0,0,0) dengan radius 5. Dengan iterasi total 250. Pertama, *quadrotor* akan berkelompok pada (0,0,0), kemudian setelah iterasi ke 50, mereka akan mencoba berpindah pada (0,0,-8) sambil menghindari tabrakan pada masing-masing anggota koloni. Hasil pengujian pertama dapat dilihat pada Gambar 2.19. Berdasarkan Gambar 2.19 dapat diketahui bahwa koloni dengan algoritma SPH dapat mengikuti kontur dari wadah, tidak seperti pada algoritma APF yang gagal mengadaptasi bentuk dari wadah.



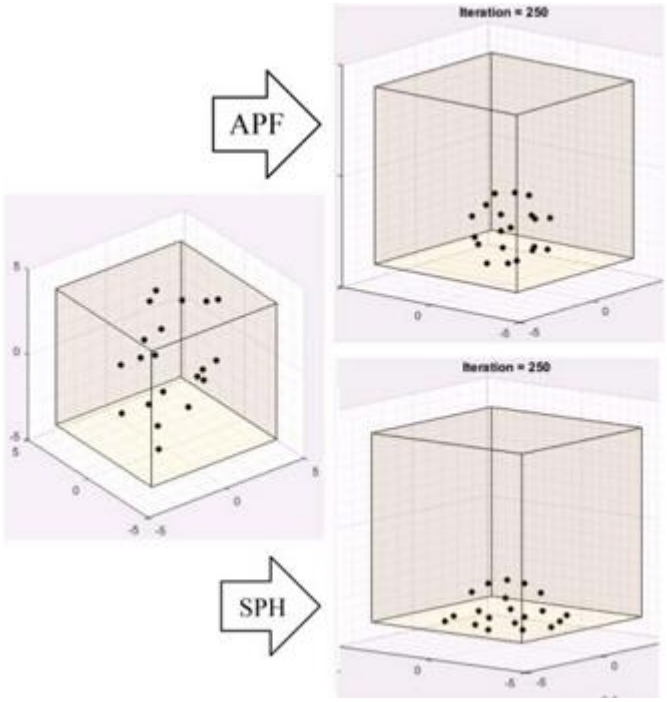
Gambar 2. 19 Pengujian pada *Container* Berbentuk Bola

Observasi lain adalah peninjauan error *centroid* pada gambar 2.20 menujukkan bahwa penggunaan algoritma SPH memiliki error lebih kecil dibandingkan dengan metode APF.



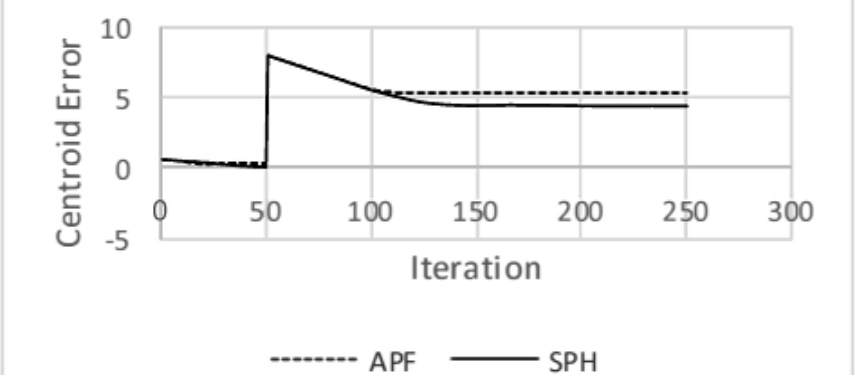
Gambar 2. 20 Peninjauan Error *Centroid* pada *Container* Berbentuk Bola

Pengujian kedua dengan wadah yang berbentuk kubus dengan titik pusat (0,0,0) dengan posisi normal berjarak 4 dari titik pusat dan panjang rusuk kubus 8.



Gambar 2. 21 Pengujian pada Wadah Berbentuk Kubus

Berdasarkan informasi pada Gambar 2.21, diketahui bahwa algoritma SPH mampu merubah formasi dengan menyesuaikan bentuk wadah, tidak seperti algoritma APF yang gagal merubah bentuk formasi sesuai karakter zat cair yang tersebar merata pada permukaan bawah suatu wadah. Hasil pengujian error *centroid* untuk wadah yang berbentuk kubus ditunjukkan pada Gambar 2.22



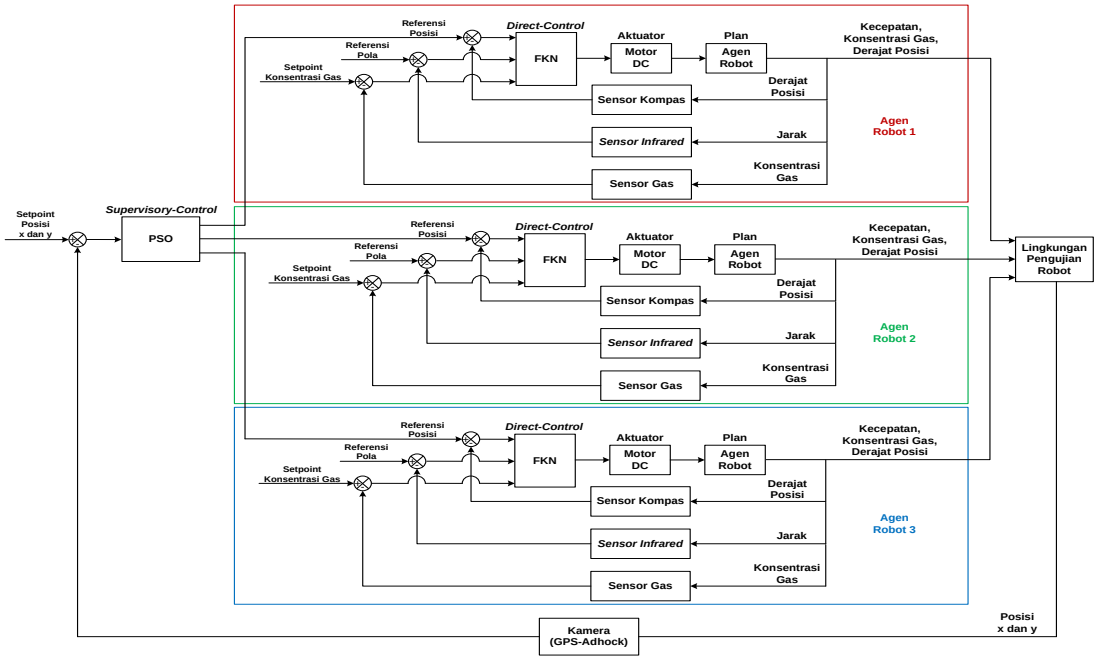
Gambar 2. 22 Peninjauan Error *Centroid* pada Wadah yang Berbentuk Kubus

Seperti pada pengujian pertama, dapat diketahui bahwa nilai error *centroid* pada algoritma SPH lebih kecil jika dibandingkan nilai error *centroid* pada algoritma APF. Pada paper ini memperkenalkan algoritma SPH untuk mendemonstrasikan perilaku agregasi *quadrotor* secara simulasi dan implementasi. Algoritma *benchmarking*/APF digunakan sebagai pembanding dari metode SPH yang digunakan. Pada pengujian dengan wadah berbentuk bola, SPH menunjukkan performa yang lebih baik daripada menggunakan algoritma APF dengan margin 0.703. Sedangkan pada wadah yang berbentuk kubus, SPH menunjukkan performa yang lebih baik daripada APF dengan margin 1.016. Dari sini, dapat dikatakan bahwa algoritma SPH lebih baik jika dibandingkan dengan algoritma APF.

Pada implementasi secara nyata diperoleh data, sistem dapat mencapai akurasi jarak 96.93% antar *quadrotor* setelah 15 detik. Hal ini berkaitan dengan nilai error 3.95 cm. Selain itu, setelah 15 detik, sistem dapat mencapai akurasi pergerakan sebesar 91.14% dengan nilai error 11.41 cm. secara umum dapat dinyatakan bahwa kesatuan dalam koloni berhasil diterapkan pada *quadrotor*.

2.1.4 Implementasi Navigasi *Multiple Autonomous Mobile Robot* untuk Mencari Sumber Gas dengan Menggunakan Metode FKN-PSO (*Fuzzy Kohonen Network - Particle Swarm Optimization*) [9]

Adapun tujuan khusus dari digunakannya metode FKN-PSO, yakni dengan menggunakan FKN maka robot dapat bernavigasi melalui pengenalan terhadap pola lingkungan yang telah ditanamkan, dan dengan menggunakan PSO robot dapat berkordinasi antar robot untuk menghindari terjadinya tabrakan sesama robot serta dapat memberikan informasi mengenai kemungkinan posisi target sumber kebocoran gas kepada seluruh robot. Penggabungan ini memiliki tujuan untuk mengoptimalisasikan navigasi robot melalui AI dan mengawasi dan mengatur pergerakan setiap robot dalam kelompok melalui PSO. Adapun diagram blok sistem pada penelitian [9] ditunjukkan pada Gambar 2.23.



Gambar 2. 23 Diagram Blok Mekanisme Kontrol FKN-PSO

Berdasarkan Gambar 2.23 pada bagian mekanisme kontrol FKN, diketahui bahwa nilai jarak diperoleh dari pembacaan sensor *infrared*. Nilai jarak tersebut dikonversikan menjadi bobot, kemudian pola bobot tersebut akan dibandingkan dengan referensi pola yang telah ditanamkan dalam FKN. Hasil dari metode FKN akan memberikan nilai kecepatan bagi aktuator motor DC untuk bergerak menghindari halangan pada lingkungan. Akan tetapi ketika sensor gas telah mendeteksi nilai *setpoint* awal untuk pendeteksian gas maka kecepatan motor DC akan diatur sedemikian untuk mengarahkan agen robot ke arah target melalui mekanisme kontrol PSO.

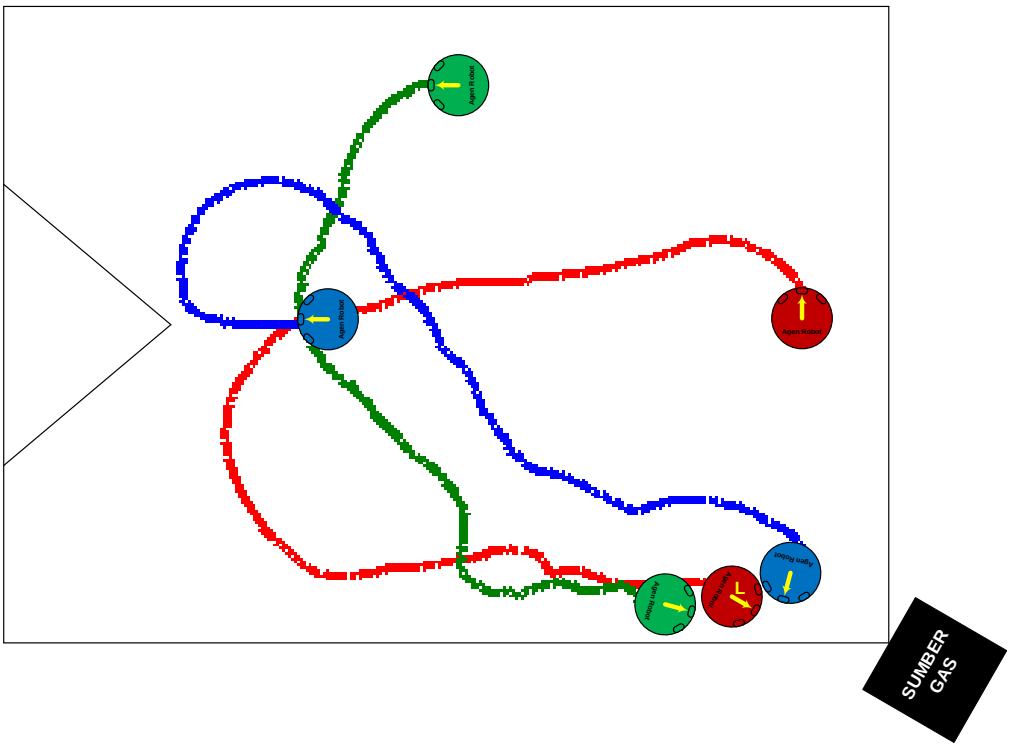
Mekanisme kontrol PSO mengendalikan posisi dari setiap agen robot menuju target sumber gas dengan referensi utama berasal dari posisi agen robot yang memiliki nilai pbest terbesar atau disebut gbest. PSO membandingkan nilai kecepatan yang seharusnya dengan nilai kecepatan hasil output dari metode FKN agar agen robot bisa mengarah menuju target. Posisi masing-masing agen robot diperoleh melalui kamera (GPS-*adhock*) sedangkan derajat antara agen robot dengan target diperoleh dari data pembacaan sensor kompas yang dikirimkan oleh agen robot. Agen robot yang pertama kali (leader) menemukan batas maksimum

konsentrasi gas (sumber gas) akan berhenti pada posisi tersebut kemudian diikuti oleh agen robot lainnya.

Pengujian dilakukan untuk mengetahui kinerja dari sistem navigasi secara keseluruhan dengan menggunakan *multiple robot*. Kinerja yang dimaksud disini adalah waktu yang dibutuhkan agen robot dalam menemukan target, tingkat konsentrasi gas yang dideteksi dan keberhasilan seluruh agen robot dalam mencapai target. Pengujian dilakukan pada 3 bentuk lingkungan yang berbeda dan dimaksudkan sebagai tingkat kesulitan lingkungan itu sendiri. Lingkungan I dengan tingkat kesulitan rendah, lingkungan II dengan tingkat kesulitan sedang dan lingkungan III sebagai tingkat kesulitan tinggi. Pengujian dilakukan sebanyak 5 kali pada masing-masing lingkungan untuk mendapatkan waktu rata-rata pencarian target dan membuktikan kehandalan metode kontrol yang digunakan.

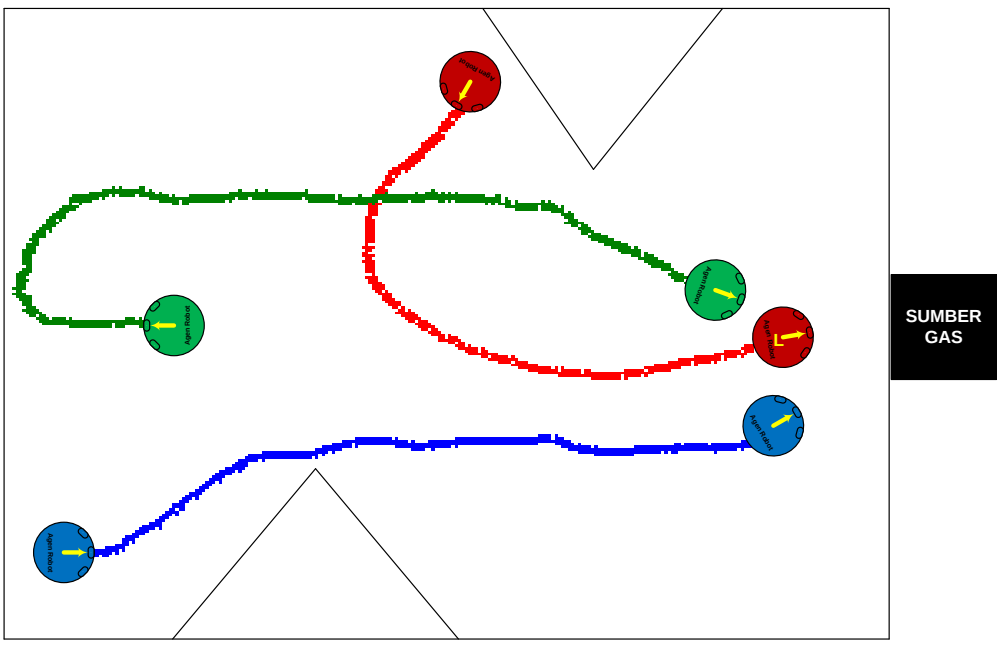
Bentuk lingkungan I bisa dikatakan memiliki tingkat kesulitan rendah bila dibandingkan dengan bentuk lingkungan lainnya. Lingkungan berbentuk persegi panjang besar dengan sebuah halangan berbentuk segitiga berukuran sedang. Agen robot diletakkan secara acak, namun titik kordinat awal posisi robot telah dicatat sebelumnya agar bisa diulangi lagi untuk pengujian selanjutnya. Target berupa sumber gas diletakkan di luar sisi sebelah kiri lingkungan dan berada di pojok bawah.

Berdasarkan hasil trajektori yang diperoleh dari Gambar 2.24, diketahui ketiga robot berhasil menemukan posisi target sumber gas. Dari trajektori yang dihasilkan juga diketahui bahwa yang menjadi *leader* pada saat menemukan target adalah robot merah, kemudian diikuti oleh robot hijau kemudian robot biru.



Gambar 2. 24 Hasil Trajektori Lingkungan I

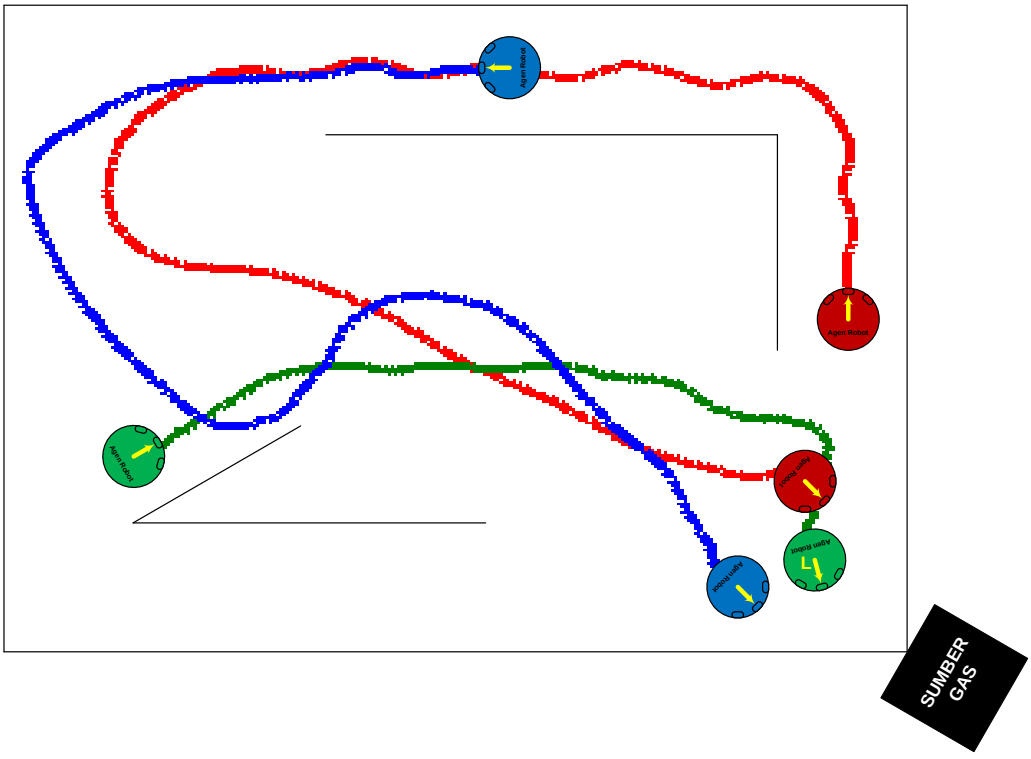
Bentuk lingkungan II memiliki tingkat kesulitan sedang bila dibandingkan dengan bentuk lingkungan lainnya. Lingkungan berbentuk persegi panjang besar dengan dua buah halangan berbentuk segitiga berukuran sedang.



Gambar 2. 25 Hasil Trajektori Lingkungan II

Dari trajektori yang dihasilkan pada lingkungan II diketahui bahwa yang menjadi *leader* saat menemukan target adalah robot merah, kemudian diikuti oleh robot biru kemudian robot hijau. Dapat dilihat juga bahwa robot lebih mengutamakan bernavigasi menghindari halangan kemudian mencari posisi target sumber gas.

Bentuk lingkungan III memiliki tingkat kesulitan tertinggi bila dibandingkan dengan bentuk lingkungan lainnya. Lingkungan memiliki bentuk yang lebih kompleks dan bentuk halangan yang lebih beragam. Agen robot juga diletakkan secara acak, namun titik kordinat awal posisi robot telah dicatat sebelumnya agar bisa diulangi lagi untuk pengujian selanjutnya. Target berupa sumber gas diletakkan di luar sisi sebelah kiri lingkungan dan dan berada di pojok bawah.



Gambar 2. 26 Hasil Trajektori Lingkungan III

Berdasarkan hasil trajektori yang diperoleh dari Gambar 2.26, diketahui ketiga robot berhasil menemukan posisi target sumber gas. Dari trajektori yang

dihasilkan juga diketahui bahwa yang menjadi *leader* adalah robot hijau, kemudian diikuti oleh robot merah kemudian robot biru.

2.1.5 Pengembangan *Adaptive Particle Swarm Optimization* (PSO) dan Aplikasinya pada Perencanaan Jalur *Mobile Robot* dengan Halangan Dinamis [10]

Perencanaan jalur berbasis metode optimasi heuristik dikembangkan untuk menyederhanakan permasalahan perencanaan jalur menjadi permasalahan optimasi. Salah satu metode optimasi heuristik yang sering digunakan dalam perencanaan jalur adalah *Particle Swarm Optimization* (PSO) karena kesederhanaan pada algoritmanya, mudah diimplementasikan dan memiliki sedikit parameter untuk diatur. Akan tetapi pada permasalahan perencanaan jalur yang kompleks dengan lingkungan dinamis, algoritma dasar PSO tidak dapat menjamin menemukan solusi optimal (*local optimum*) dikarenakan konvergensi prematur yang menyebabkan terjadi tumbukan dengan halangan dan jalur yang lebih panjang.

Pada penelitian ini setelah perilaku pencarian PSO dianalisa, PSO adaptif dikembangkan dengan menggunakan fungsi Gaussian dalam pengaturan nilai parameter pada PSO untuk mempercepat konvergensi dan reinisialisai partikel dilakukan untuk mencegah terjadinya konvergensi prematur.

Simulasi dan perbandingan dengan algoritma *Adaptif Inertia* (AIW) PSO dan standard PSO menunjukan algoritma yang diusulkan dapat menemukan solusi optimal lebih cepat dengan konvergensi kurang dari 150 iterasi pada halangan statis dan 200 iterasi pada halangan bergerak. Selain itu algoritma yang diusulkan memiliki 3% panjang lintasan yang lebih pendek, 10% lebih *smooth* dan lebih terjamin terhindar dari tumbukan.

2.2 Teori Dasar

2.2.1 *Particle Swarm Optimization* [5]

PSO didasarkan pada perilaku sebuah kawasan burung atau ikan. Algoritma PSO meniru perilaku sosial organisme ini. Perilaku sosial terdiri dari tindakan individu dan pengaruh dari individu-individu lain dalam sesuatu

kelompok, kata partikel menunjukkan misalnya, seekor burung dalam kawanan burung. setiap individu atau partikel berperilaku dengan cara menggunakan kecerdasannya sendiri dan juga dipengaruhi perilaku kelompoknya. Dengan demikian, jika satu partikel atau seekor burung menemukan jalan yang tepat atau pendek menuju ke sumber makanan, sisa kelompok yang lain juga akan dapat segera mengikuti jalan tersebut meskipun lokasi mereka jauh dari kelompok tersebut.

Dalam *particle swarm optimization* (PSO), kawan diasumsikan mempunyai ukuran tertentu dengan posisi awal setiap partikel bersifat acak dalam ruang multidimensi. Setiap partikel diasumsikan dua karakteristik yaitu, posisi dan kecepatan. Setiap partikel dalam ruang bergerak dalam posisi tertentu dan mengingat posisi terbaik yang dilalui atau ditemukan terhadap sumber makanan atau nilai fungsi objektif. Setiap partikel menyampaikan informasi atau posisi terbaiknya kepada partikel yang lain dan menyesuaikan posisi dan kecepatan masing-masing berdasarkan informasi yang diterima mengenai posisi tersebut. Meskipun setiap burung memiliki keterbatasan dalam hal kecerdasan, biasanya ia akan mengikuti kebiasaan seperti berikut:

1. Seekor burung tidak akan terlalu dekat dengan burung yang lain
2. Burung tersebut akan mengarahkan terbangnya ke arah rata-rata keseluruhan burung.
3. Akan memposisikan diri dengan rata-rata posisi burung yang lain dengan menjaga jarak antar burung dalam kawanan itu sehingga tidak terlalu jauh.

Dengan demikian perilaku kawanan burung didasarkan dari kombinasi tiga faktor sebagai berikut:

1. Kohesi - terbang bersama
2. Separasi – jangan terlalu dekat
3. Penyesuaian (*alignment*) – mengikuti arah bersama

Jadi PSO dikembangkan dengan berdasarkan pada model berikut:

1. Kriteria seekor burung mendekati target atau makanan (atau bisa minimum atau maksimum suat fungsi tujuan) secara cepat mengirim informasi kepada burung-burung yang lain dalam kawasan tertentu

2. Burung yang lain akan mengikuti arah menuju ke makan tetapi tidak secara langsung
3. Ada komponen yang tergantung pada pikiran setiap burung, yaitu memorinya tentang apa yang sudah dilewat sebelumnya.

Pada algoritma PSO ini, pencarian solusi dilakukan oleh suat populasi yang terdiri dari beberapa partikel. Populasi dibangkitkan secara *random* dengan batasan permasalahan yang dihadapi. Setiap partikel mempresentasikan partikel atau solusi dari permasalahan yang dihadapi. Setiap partikel melakukan pencarian solusi yang optimal dengan melintasi ruang pencarian. Hal ini dilakukan dengan cara setiap partikel melakukan penyesuain terhadap posisi terbaik dari setiap partikel tersebut (*local best*) dan posisi partikel terbaik dari seluruh kawanan (*global best*) selama melintasi ruang pencarian. Jadi penyebaran pengalaman atau informasi terjadi dalam partikel itu sendiri dan antara suatu partikel dengan partikel terbaik dari seluruh kawanan selama proses pencarian solusi. Setelah itu, dilakukan proses pencarian untuk mencari posisi terbaik setiap partikel dalam jumlah iterasi tertentu sampai didapatkan posisi relatif yang *steady* atau mencapai batas iterasi yang telah ditetapkan. Pada setiap iterasi, setiap solusi yang direpresentasikan oleh posisi partikel, dievaluasi performanya dengan cara memasukan solusi tersebut ke dalam *fitness function*.

Setiap partikel diperlakukan seperti titik pada suatu dimensi ruang tertentu kemudian terdapat dua faktor yang memberikan karakter terhadap status partikel pada ruang pencarian yaitu posisi partikel dan kecepatan partikel. Persamaan 2.5 - 2.6 merupakan formulasi matematika yang menggambarkan posisi dan kecepatan partikel suatu dimensi ruang tertentu,

$$X_i(t) = x_{i`0}(t), x_{i1}(t), ..., x_{iN}(t) \quad (2.5)$$

$$V_i(t) = v_{i`0}(t), v_{i1}(t), ..., v_{iN}(t) \quad (2.6)$$

Dengan,

X = posisi partikel

V = kecepatan partikel

I = indeks partikel

t = iterasi ke-t

N = ukuran dimensi ruang

Persamaan 2.7-2.8 merupakan model matematika yang menggambarkan mekanisme updating status partikel Kennedy dan Eberhert:

$$V_i(t) = v_{i`}(t-1) + c_1 r_1 (X_i L - X_i(t-1)) + c_2 r_2 (X_i G - X_i(t-1)) \quad (2.7)$$

$$X_i(t) = V_{i`}(t) + X_i(t-1) \quad (2.8)$$

dengan,

X_i = *local best* dari partikel ke – i

$X_i G$ = *global best* dari seluruh kawanan

c_1 = *learning factor* c2 = *learning factor*

r_1 = bilangan random yang bernilai antara 0 sampai 1

r_2 = bilangan random yang bernilai antara 0 sampai 1

Persamaan 2.7 digunakan untuk menghitung kecepatan partikel yang baru berdasarkan kecepatan sebelumnya, jarak antara posisi saat ini dengan posisi terbaik partikel (*local best*), dan jarak antara posisi saat ini dengan posisi terbaik kawanan (*global best*). Kemudian partikel terbang menuju posisi yang baru berdasarkan Persamaan 2.8. Setelah algoritma PSO ini dijalankan dengan sejumlah iterasi tertentu hingga mencapai kriteria pemberhentian, maka akan didapatkan solusi yang terletak pada *global best*.

Algoritma PSO meliputi langkah berikut:

1. Membangkitkan posisi awal sejumlah partikel sekaligus kecepatan awalnya secara *random*.
2. Mengevaluasi *fitness* dari masing-masing partikel berdasarkan posisinya.
3. Menentukan partikel dengan *fitness* terbaik dan tetapkan sebagai G_{best}. Untuk setiap partikel G_{best} awal sama dengan posisi awal.
4. Mengulangi langkah berikut sampai pemberhentian kriteria dipenuhi
5. Menggunakan P_{best} dan G_{best} yang ada, perbarui kecepatan setiap partikel menggunakan Persamaan 2.7. lalu dengan kecepatan baru

yang didapat, perbarui posisi setiap partikel menggunakan Persamaan 2.8.

6. Mengevaluasi *fitness* setiap partikel.
7. Menentukan partikel dengan *fitness* terbaik, dan tetapkan sebagai G_{best}. Untuk setiap partikel tentukan P_{best} dengan membandingkan posisi sekarang dengan P_{best} dari iterasi sebelumnya.
8. Mengecek pemberhentian kriteria bila dipenuhi berhenti, bila sebaliknya kembali ke langkah 1.

2.2.2 *Artificial Neural Network* (ANN)

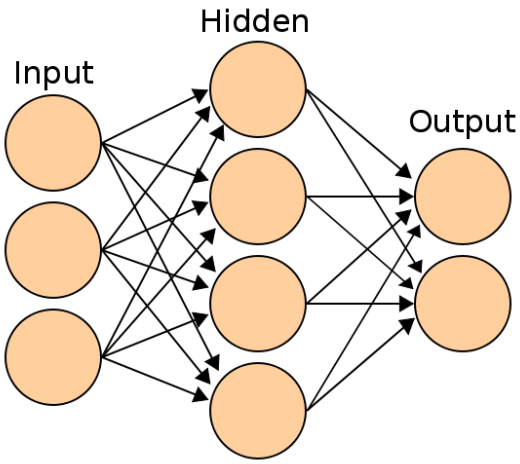
Jaringan saraf tiruan (JST) (Bahasa Inggris: *artificial neural network* (ANN), atau juga disebut *simulated neural network* (SNN), atau umumnya hanya disebut *neural network* (NN), adalah jaringan dari sekelompok unit pemroses kecil yang dimodelkan berdasarkan sistem saraf manusia. JST merupakan sistem adaptif yang dapat mengubah strukturnya untuk memecahkan masalah berdasarkan informasi eksternal maupun internal yang mengalir melalui jaringan tersebut. Oleh karena sifatnya yang adaptif, JST juga sering disebut dengan jaringan adaptif.

Secara sederhana, JST adalah sebuah alat pemodelan data statistik non-linier. JST dapat digunakan untuk memodelkan hubungan yang kompleks antara *input* dan *output* untuk menemukan pola-pola pada data. Menurut suatu teorema yang disebut "teorema penaksiran universal", JST dengan minimal sebuah lapis tersembunyi dengan fungsi aktivasi *nonlinear* dapat memodelkan seluruh fungsi terukur boreal apapun dari suatu dimensi ke dimensi lainnya [6].

Model pada JST pada dasarnya merupakan fungsi model matematika yang mendefinisikan fungsi $f : X \rightarrow Y$. Istilah "jaringan" pada JST merujuk pada interkoneksi dari beberapa neuron yang diletakkan pada lapisan yang berbeda. Secara umum, lapisan pada JST dibagi menjadi tiga bagian:

- Lapis masukan (*input layer*) terdiri dari neuron yang menerima data masukan dari variabel *X*. Semua neuron pada lapis ini dapat terhubung ke neuron pada lapisan tersembunyi atau langsung ke lapisan luaran jika jaringan tidak menggunakan lapisan tersembunyi.

- Lapisan tersembunyi (*hidden layer*) terdiri dari neuron yang menerima data dari lapisan masukan.
- Lapisan luaran (*output layer*) terdiri dari neuron yang menerima data dari lapisan tersembunyi atau langsung dari lapisan masukan yang nilai luarannya melambangkan hasil kalkulasi dari *X* menjadi nilai *Y*.



Gambar 2. 27 Model Jaringan Syaraf Tiruan

Secara matematis, neuron merupakan sebuah fungsi yang menerima masukan dari lapisan sebelumnya $g_i(x)$ (lapisan ke-i). Fungsi ini pada umumnya mengolah sebuah vektor untuk kemudian diubah ke nilai skalar melalui komposisi nonlinear *weighted sum*, dimana $f(x) = K\left(\sum w_i g_i(x)\right)$, K merupakan fungsi khusus yang sering disebut dengan fungsi aktivasi dan w merupakan bobot atau *weight* [6]. Model ANN ditunjukkan pada Gambar 2.23

2.2.3 *Karakteristik Tingkah Laku dan Motifasi Kecerdasan Swarm*

Umumnya semua hal yang terkait dengan kecerdasan *swarm* berasal dari kelompok atau perilaku sosial hewan maupun serangga. Hewan ini secara alami tidak terorganisir tetapi semata-mata tergantung pada tindakan individu lainnya. Berdasarkan komunikasi dengan anggota kawanan terdekat dan tindakan yang telah dilakukan, individu dapat mengevaluasi dan kemudian menghasilkan perilaku lain yang akan berkontribusi pada tujuan kelompok tersebut. Semua anggota *swarm*

mengolah semua data yang diambil dari lingkungan tersebut, sehingga setiap aktivitas individual yang sedang berlangsung akan dipengaruhi oleh lingkungannya.

1. Pengelompokan (*Aggregation*)

   Agregasi merupakan karakteristik dasar alamiah suatu *swarm*. Dimana *swarm* secara alamiah akan bergerak baik berkelompok ataupun secara individu untuk tujuan yang sama (mencari target/sumber makanan). agregasi adalah salah satu perilaku yang tidak bisa dihindari dalam sistem *swarm robotic*. Perilaku ini mempunyai kemampuan masing-masing individu untuk berkomunikasi atau menganalisis tugas jika anggota kawanan berada dalam jarak dekat. Biasanya perilaku ini akan dikombinasikan dengan perilaku *swarm* lainnya untuk mencapai tujuan yang ingin dicapai.

2. Proses Mencari Sumber Makanan

   Perilaku biologis lainnya pada *swarm* yang dapat diadopsi dalam robotika *swarm* adalah dalam hal mencari makanan. Perilaku ini erat hubungannya dengan agregasi yang merupakan reaksi kolektif dari individu berdasarkan lingkungannya, sehingga lingkungan merupakan faktor besar dalam menghasilkan perilaku dalam kawanan. Di sini lingkungan *swarm* terbagi menjadi dua jenis lingkungan yaitu wilayah yang menguntungkan dan wilayah yang berbahaya. Wilayah ini serupa dengan wilayah yang tersedia sumber makanan dan wilayah yang berpredator, wilayah yang menguntungkan wilayah dimana makanan tersedia melimpah untuk kawanan sebagai lokasi yang dituju kawanan sementara wilayah berbahaya diwakili oleh daerah dimana terdapat predator yang dapat memangsa kawanan dan perlu dihindari yang dapat dikatakan sebagai *obstacle*.

3. Formasi Terbang

   Formasi terbang merupakan aksi dari proses mencari makan dimana akan tercipta konfigurasi geometris tertentu dari individu yang awalnya tidak terorganisir. Penyebaran konfigurasi yang dibuat, sangat tergantung pada

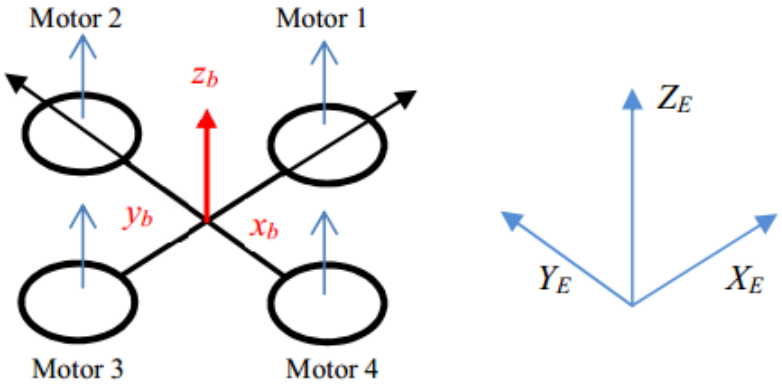
batasan pergerakan masing-masing individu sambil menjaga konfigurasi geometrik yang telah direncanakan. Kawanan biasanya mengasah kemampuan dalam berbagai formasi, misalnya formasi yang dihasilkan oleh sekelompok burung dalam menghindari predator dan cara menangkap mangsa. Keunikan dari hewan ini adalah menggunakan formasi penerbangan untuk menghemat energi atau mengurangi usaha lainnya yang dilakukan individu dalam terbang. Dalam formasi mereka menggunakan pemimpin yang berfungsi sebagai penyerap halangan jika anggota kawanan lainnya tidak mengalami halangan yang sama.

4. Swarm Tracking

Salah satu unsur pembentukan formasi terbang dan agregasi adalah *swarm tracking*. Terdapat contoh dimana target, baik mangsa atau pemangsa secara bersamaan bergerak dan berinteraksi di dalam kawanan. Target tersebut mungkin memiliki tujuan untuk menangkap kawanan atau menghindari kawanan. Kawanan tersebut perlu mempertahankan formasi saat melacak target. Formasi ini digunakan untuk membidik target saat terbang, yang dapat dianalogikan seperti bantuan informasi menuju target dalam terbang dari suatu titik ke titik lainnya. Sebaliknya sebuah kawanan dapat melacak target dan terus menyusun ulang formasinya untuk menghindari target yang dilacak yang selanjutnya memungkinkan kawanan untuk menghindari rintangan sepanjang lintasan yang dilaluinya.

2.2.4 Sistem Koordinat *Quadrotor*

Sistem koordinat digunakan dalam pemodelan *quadrotor*. Sistem koordinat pada *quadrotor* terdiri dari *E* (*O, X, Y, Z*) sebagai *inertial frame* dan *b* (*o', x, y, z*) sebagai *body coordinate* seperti yang diilustrasikan pada Gambar 2.24,



Gambar 2. 28 Sistem Koordinat *Quadrotor*

Untuk mengubah vektor *state* dari *inertial frame* (*E*) ke *body coordinate* (*B*) diperlukan matrik transformasi. Matrik ini menjelaskan tentang konversi rotasi yang pertama terhadap sumbu *x*, kemudian terhadap sumbu *y*, dan yang terakhir terhadap sumbu *z*. Matrik tersebut dapat dicari menggunakan Persamaan (2.9).

$$ {}^{E}_{B}R = R_{x,\phi} * R_{y,\theta} * R_{z,\psi} \tag{2.9} $$

di mana, $R_{x,\phi}, R_{y,\theta}$ dan $R_{z,\Psi}$ merupakan matrik rotasi pada setiap sumbunya, seperti yang dijabarkan sebagai berikut:

- Rotasi terhadap sumbu $x : R_{x,\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$

dengan sudut *roll* $\phi \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$

- Rotasi terhadap sumbu $y : R_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$

dengan sudut *pitch* $\theta \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$

Rotasi terhadap sumbu $z : R_{z,\psi} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$

dengan sudut *yaw* $\psi \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$

Sehingga diperoleh matrik tranformasi dari *inertial frame* ke *body coordinate*

$$ {}^{E}_{B}R = R_{x,\phi} * R_{y,\theta} * R_{z,\psi} = \begin{bmatrix} C\theta C\psi & -C\theta S\psi & S\theta \\ S\phi S\theta C\psi + C\phi S\psi & -S\phi S\theta S\psi + C\phi C\psi & -S\phi C\theta \\ -C\phi S\theta C\psi + S\phi S\psi & C\phi S\theta S\psi + S\phi C\psi & C\phi C\theta \end{bmatrix} $$

Sedangkan matrik transformasi dari *body coordinate* (*B*) ke *inertial frame* (*E*) seperti yang ditunjukkan pada Persamaan (2.10)

$$ {}^{B}_{E}R = R_{z,\psi} * R_{y,\theta} * R_{x,\phi} = \begin{bmatrix} C\theta C\psi & -C\phi S\psi + S\phi S\theta C\psi & S\phi S\psi + C\phi S\theta C\psi \\ C\theta S\psi & C\phi C\psi + S\phi S\theta S\psi & -S\phi C\psi + C\phi S\theta S\psi \\ -S\theta & S\phi C\theta & C\phi C\theta \end{bmatrix} \quad (2.10) $$

2.2.5 Dasar Pemodelan *Quadrotor*

Pemodelan secara fisik terbilang kompleks apabila tanpa adanya asumsi yang digunakan untuk menyederhanakan persamaan. Beberapa asumsi yang digunakan dalam pemodelan ini adalah

1. Percepatan gravitasi konstan dan tegak lurus terhadap permukaan bumi
2. Posisi pusat massa tepat di tengah
3. Struktur quadrotor pejal dan simetris
4. Struktur propeller adalah pejal

Quadrotor memiliki 6 *degree of freedom* (DoF) dengan 12 *output*, *output* tersebut menghasilkan gerakan yang mempresentasikan *attitude* dari *quadrotor* yaitu gerakan translasi *x, y, z* dan gerakan rotasi ϕ, θ, ψ. Variabel-variabel tersebut ditunjukkan pada Tabel 2.3.

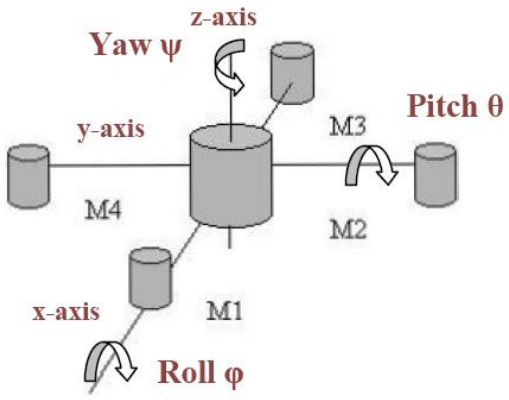
Tabel 2. 3 Variabel pada Pergerakan *Quadrotor*

Variabel	Keterangan
x	Posisi *quadrotor* terhadap sumbu X_E
y	Posisi *quadrotor* terhadap sumbu Y_E
z	Posisi *quadrotor* terhadap sumbu Z_E
u	Kecepatan *quadrotor* yang diukur pada sumbu x_b
v	Kecepatan *quadrotor* yang diukur pada sumbu y_b
w	Kecepatan *quadrotor* yang diukur pada sumbu z_b
ϕ	Sudut *roll* terhadap sumbu X_E
θ	Sudut *pitch* terhadap sumbu Y_E
ψ	Sudut *yaw* terhadap sumbu Z_E
p	Kecepatan sudut *roll* yang diukur pada sumbu x_b
q	Kecepatan sudut *pitch* yang diukur pada sumbu y_b
r	Kecepatan sudut *yaw* yang diukur pada sumbu z_b

2.2.6 Inersia *Quadrotor*

Ada beberapa asumsi yang diberikan sebelum menghitung momen inersia terhadap sumbu X_b dan Y_b dari *quadrotor*. Asumsi-asumsi tersebut yaitu

1. Motor M_1 dan M_2 berbentuk silindris dengan jari-jari r, tinggi h, dan massa m.
2. Badan tengah dari *quadrotor* juga berbentuk silindris dengan jari-jari R, tinggi H, dan *massa* M.



Gambar 2. 29Inersia pada Sumbu X_b, Y_b dan Z_b

Pada Gambar 2.7 menampilkan bentuk silindris yang menjadi acuan menghitung inersia *quadrotor*. Momen inersia terhadap sumbu *x* ada 2 bagian, yaitu:

1. Hubungan gaya dari Motor M_2 dan M_4 terhadap sumbu *X* dengan jari-jari putaran l
2. Hubungan Motor M_1, M_3, dan *body* tengah terhadap sumbu X_b

Momen inersia bentuk silindris yang tegak lurus terhadap badan *quadrotor* terlihat pada Persamaan (2.11). Bagian pertama, hubungan gaya dari motor M_2 dan M_4 terhadap sumbu X_b ditampilkan pada Persamaan (2.12). Bagian kedua, hubungan motor M_1, M_3, dan *body* tengah terhadap sumbu X_b ditampilkan pada Persamaan (2.13) dan (2.14).

$$J_{c,x} = \frac{m_c * r_c^{\,2}}{4} + \frac{m_c h_c^{\,2}}{12} \tag{2.11}$$

$$J_{2-4,x} = 2m_r l^2 \tag{2.12}$$

$$J_{1,x} = \frac{m_1 * r_1^{\,2}}{4} + \frac{m_1 h_1^{\,2}}{12} \tag{2.13}$$

$$J_{3,x} = \frac{m_3 * r_3^{\,2}}{4} + \frac{m_3 h_3^{\,2}}{12} \tag{2.14}$$

Jika diketahui bahwa,

$m_1 = m_2 = m_3 = m_4 = m_r$ (2.15)

$r_1 = r_2 = r_3 = r_4 = r_r$ dan $h_1 = h_2 = h_3 = h_4 = h_r$

Maka total persamaan J_{xx} dan J_{yy} merupakan hasil penjumlahan dari Persamaan (2.13) dan (2.14) yang dinyatakan dalam Persamaan (2.16) dan (2.17).

$$J_{xx} = \frac{m r_r^{\,2}}{2} + \frac{m h_r^{\,2}}{6} + \frac{m_o r_c^{\,2}}{4} + \frac{m_o h_c^{\,2}}{12} + 2m_r l^2 \tag{2.16}$$

$$J_{yy} = \frac{m r_r^{\,2}}{2} + \frac{m h_r^{\,2}}{6} + \frac{m_o r_c^{\,2}}{4} + \frac{m_o h_c^{\,2}}{12} + 2m_r l^2 \tag{2.17}$$

Untuk menghitung momentum inersia pada sumbu Z_b, dapat dibagi menjadi 2 bagian, yaitu

1. Momen inersia pada badan tengah *quadrotor*
2. Hubungan motor M_1, M_2, M_3, dan M_4

Bagian pertama, momentum inersia pada badan tengah *quadrotor* adalah terlihat pada Persamaan (2.18). Bagian kedua, hubungan motor M_1, M_2, M_3, dan M_4 terlihat pada Persamaan (2.19).

$$J_{c-z} = \frac{m_c r_c^{\,2}}{2} \tag{2.18}$$

$$J_{1-2-3-4,\,z} = 4 m_r l^2 \tag{2.19}$$

Total persamaan J_{zz} merupakan penjumlahan dari Persamaan (2.18) dan (2.19) yang ditunjukkan pada Persamaan (2.20).

$$J_{zz} = \frac{m_c r_c^{\,2}}{2} + 4 m_r l^2 \tag{2.20}$$

2.2.7 Gaya dan Momen *Quadrotor*

Pada bagian ini akan dibahas mengenai gaya dan momen yang bekerja pada *quadrotor*. Pada *quadrotor* dianggap tidak ada yang berbentuk aerodinamis sehingga gaya aerodinamis dan momen aerodinamis yang terjadi dapat diabaikan. Berdasarkan gaya yang terjadi pada tiap motor, dapat dihitung persamaan torsi yang terjadi pada gerak *roll*, *pitch*, dan *yaw*.

$$U_1 = F_{T1} + F_{T2} + F_{T3} + F_{T4} \tag{2.21}$$

$$U_2 = F_{T2} - F_{T4} \tag{2.22}$$

$$U_3 = F_{T1} - F_{T3} \tag{2.23}$$

$$U_4 = d(F_{T1} + F_{T3} - F_{T2} - F_{T4}) \tag{2.24}$$

dengan d adalah konstanta *drag* yang terjadi pada *quadrotor* dan F_{Ti} merupakan konstanta *thrust* pada setiap baling-baling yang dinyatakan dalam Persamaan (2.25).

$$F_{Ti} = K \frac{\omega}{s + \omega} u_i \tag{2.25}$$

2.2.8 Model Dinamik Gerak Translasi *Quadrotor*

Dinamika gerak translasi pada *quadrotor* diperoleh berdasarkan persamaan hukum Newton-II, sehingga diperoleh persamaan

$$\Sigma F = m\ddot{\xi} \tag{2.26}$$

$$F_f + F_t + F_g = m\ddot{\xi} \tag{2.27}$$

Jika gaya gesek diabaikan maka resultan gaya yang bekerja dinyatakan dalam Persamaan (2.28).

$$F_f + F_g = m\ddot{\xi} \tag{2.28}$$

ξ merupakan posisi dari pusat *massa quadrotor* terhadap *inertial frame* yang dinyatakan pada Persamaan (2.29).

$$\xi = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.29}$$

F_f merupakan resultan gaya yang dihasilkan oleh keempat rotor yang dinyatakan pada Persamaan (2.30).

$$F_f = \begin{bmatrix} F_{fx} \\ F_{fy} \\ F_{fz} \end{bmatrix} \tag{2.30}$$

dengan,

F_{fx} : Komponen gaya dalam arah sumbu *x*

F_{fy} : Komponen gaya dalam arah sumbu *y*

F_{fz} : Komponen gaya dalam arah sumbu *z*

Berdasarkan pembahasan pada bab dua diketahui bahwa vektor transformasi dari *body coordinate* ke *inertial frame* dinyatakan oleh Persamaan (2.31).

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} C\theta C\psi & -C\phi S\psi + S\phi S\theta C\psi & S\phi S\psi + C\phi S\theta C\psi \\ C\theta S\psi & C\phi C\psi + S\phi S\theta S\psi & -S\phi C\psi + C\phi S\theta S\psi \\ -S\theta & S\phi C\theta & C\phi C\theta \end{bmatrix} \begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} \tag{2.31}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_B(C\theta C\psi) + y_B(-C\phi S\psi + S\phi S\theta C\psi) + z_B(S\phi S\psi + C\phi S\theta C\psi) \\ x_B(C\theta S\psi) + y_B(C\phi C\psi + S\phi S\theta S\psi) + z_B(-S\phi C\psi + C\phi S\theta S\psi) \\ x_B(-S\theta) + y_B(S\phi C\theta) + z_B(C\phi C\theta) \end{bmatrix} \tag{2.32}$$

dengan,

C : fungsi *cosinus*

S : fungsi *sinus*

Persamaan (2.32) menyatakan bahwa komponen dari z_B pada sumbu *x* adalah $z_B(S\phi S\psi + C\phi S\theta C\psi)$, komponen dari z_B pada sumbu *y* adalah $z_B(-S\phi C\psi + C\phi S\theta S\psi)$ dan komponen dari z_B pada sumbu *z* adalah $z_B(C\phi C\theta)$. Arah dari F_f searah dengan z_B sehingga komponen gaya total (F_f) pada sumbu *x*, *y*, dan *z* dinyatakan dalam Persamaan (2.33)-(2.35).

$$F_{fx} = F_f(S\phi S\psi + C\phi S\theta C\psi) \tag{2.33}$$

$$F_{fy} = F_f(-S\phi C\psi + C\phi S\theta S\psi) \tag{2.34}$$

$$F_{fz} = F_f(C\phi C\theta) \tag{2.35}$$

atau

$$\begin{bmatrix} F_{fx} \\ F_{fy} \\ F_{fz} \end{bmatrix} = \begin{bmatrix} (S\phi S\psi + C\phi S\theta C\psi) \\ (-S\phi C\psi + C\phi S\theta S\psi) \\ (C\phi C\theta) \end{bmatrix} * U_1 \tag{2.36}$$

F_g merupakan gaya tarik gravitasi yang dinyatakan dalam Persamaan (2.37).

$$F_g = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \tag{2.37}$$

Dinamika gerak translasi pada *quadrotor* dapat diperoleh dengan mensubstitusikan gaya F_f dan F_g kedalam persamaan hukum Newton-II, sebagaimana dinyatakan dalam Persamaan (2.38).

$$F_f + F_g = m\ddot{\xi}$$

$$\begin{bmatrix} (S\phi S\psi + C\phi S\theta C\psi) \\ (-S\phi C\psi + C\phi S\theta S\psi) \\ (C\phi C\theta) \end{bmatrix} * U_1 + \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} = m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} \tag{2.38}$$

Berdasarkan Persamaan (2.38) dapat diperoleh persamaan lengkap dinamika gerak translasi *quadrotor* yang dinyatakan dalam Persamaan (2.39)-(2.41).

$$\ddot{x} = \left(\sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi\right)\frac{U_1}{m} \tag{2.39}$$

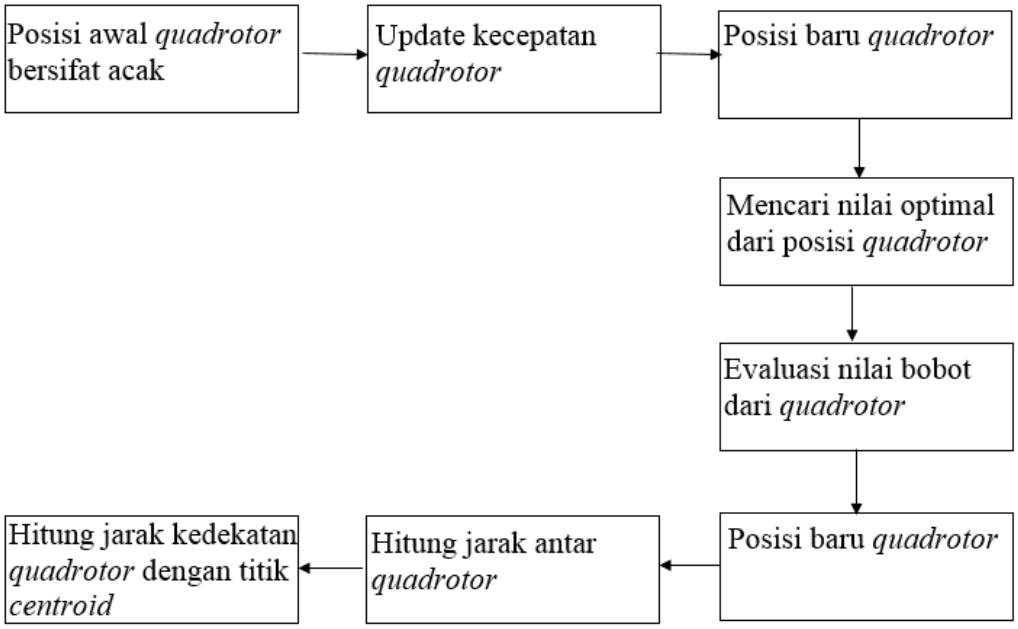
$$\ddot{y} = \left(-\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi\right)\frac{U_1}{m} \tag{2.40}$$

$$\ddot{z} = \frac{U_1}{m}\cos\phi\cos\theta - g \tag{2.41}$$

BAB 3
METODOLOGI PENELITIAN

3.1 Perancangan Metode Distribusi *Quadrotor Swarm* untuk Pelacakan Titik *Centroid*

Gambar 3.1 menggambarkan tahapan dalam perancangan metode distribusi *quadrotor swarm* untuk pelacakan titik *centroid* menggunakan metode *artificial neural network-self organization map* yang telah dimodifikasi (*Modified ANNSOM*).



Gambar 3. 1 Tahapan Perancangan Metode *Modified Artificial Neural Network-Self Organization Map* (*Modified ANNSOM*)

Adapun algortima untuk penyusuan metode *Artificial Neural Network-Self Organization Map* Termodifikasi (*Modified ANNSOM*) ditunjukkan pada Tabel 3.1.

Tabel 3. 2 Algortima untuk Penyusuan Metode *Artificial Neural Network-Self Organization Map* Termodifikasi (*Modified ANNSOM*)

1. Menentukan posisi awal dari anggota *quadrotor swarm* secara acak

   $x_{1,\ldots n} = rand()$

   $y_{1,\ldots n} = rand()$

2. Inisialisasi *weight* w_1,w_2,w_3, dan bias B_1,B_2,B_3.

3. Hitung partikel aktivasi V_{h1}, V_{h2}, V_0, dan output masing-masing neuron

   $V_h = B + \sum w \bullet x$

   Output neuron dengan fungsi aktivasi *binary sigmoid*

   $0 = \dfrac{1}{1+\exp(-V_h)}$

4. *Feedbackward* untuk *layer output*

   $f = (y_d - y) \bullet f'(v)$

5. Hitung *weight correction term*

   $\Delta w = \mu \delta_0 V_h$

6. Hitung update *weight* pada masing-masing layer

   $w_{n+1} = w + \Delta w$

7. Evaluasi *error* pada *NN*

   $SSE = (\sum errortotal)^2$

8. Setelah *weight* diperoleh maka langkah selanjutnya adalah menentukan nilai *x* dan *y* pada *NN*

   $x' = NN(x)$

   $y' = NN(y)$

9. Hitung persamaan *quadrotor* pada sumbu *x* dan sumbu *y*

   $\ddot{x} = \left(\sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi\right)\dfrac{U_1}{m}$

   $\ddot{y} = \left(-\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi\right)\dfrac{U_1}{m}$

10. Hitung kecepatan dan update kecepatan

$$V_x^{'} = \frac{(x^{'} + x_p)}{\Delta t}$$

$$V_y^{'} = \frac{(y^{'} + y_p)}{\Delta t}$$

11. Hitung update posisi berdasarkan kecepatan dan update kecepatan pada langkah 10

$$x_{new} = V_x^{'} \bullet \Delta t$$

$$y_{new} = V_y^{'} \bullet \Delta t$$

12. Jika posisi antar *quadrotor* dibawah nilai *threshold* maka nilai koordinat ditambah agar berada pada posisi terdekat dengan titik *centroid*
13. Menghitung jarak antar *quadrotor* untuk menghindari terjadinya tabrakan

$$d = |x - y| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$$

Untuk menghindari terjadinya tabrakan antar anggota *quadrotor swarm* saat melintasi jalur yang sama yaitu dengan melakukan pembelokan arah berdasarkan algoritma sebagai berikut:

1. Posisi awal anggota *quadrotor swarm* adalah acak
2. Anggota *quadrotor swarm* ke-*n* menuju ke titik *centroid*
3. Hitung nilai jarak antar *quadrotor*
4. Jika jarak Anggota *quadrotor swarm* ke-*n* dengan Anggota *quadrotor swarm* ke *n-1* < 0.2 meter, maka posisi *quadrotor* ke- *n* sama dengan posisi *quadrotor* ke *n-1* +0.3 meter.
5. Proses selanjutnya berulang pada langkah 2.

Keseluruhan proses dibatasi pada iterasi maksimal sebanyak 200. Apabila seluruh anggota *quadrotor swarm* mampu melacak posisi titik *centroid* sebelum mencapai batas iterasi maksimal, maka iterasi akan berhenti dan dinyatakan sebagai solusi optimalnya. Namun, apabila anggota *quadrotor swarm* belum mampu melacak posisi titik *centroid* sampai dengan iterasi ke-200 maka proses pelacakan

akan dihentikan pada batas iterasi maksimalnya dan dinyatakan sebagai solusi optimal dalam proses pelacakan pada kondisi tersebut.

3.2 Koordinasi dan Permasalahan Kontrol *Quadrotor Swarm*

Swarm quadrotor berada di dalam *bounded* area yang diharapkan beroperasi dengan batasan pada Γ^n. Elemen *swarm* diwakili oleh $X_i(t)$, dimana $i = 1,2,3,...,N$, yang merepresentasikan status atau *state* setiap *quadrotor* dengan jumlah *N* pada waktu *t*. Fungsi kontrol untuk setiap *quadrotor* diwakili oleh fungsi $u_i(t)$, untuk bentuk yang seragam pada setiap individu $i = 1,2,3,...,N$.

1. Pengelompokan (*Aggregation*)

   Agregasi merupakan kemampuan kawanan untuk bekerjasama dan mengumpulkan atau mengatur dirinya sendiri secara spesifik sesuai lokasi dan menghindari benturan dengan anggota kawanan lainnya. Dengan demikian, agregasi adalah salah satu perilaku yang tidak bisa dihindari dalam sistem *swarm quadrotor*. Perilaku ini mempunyai kemampuan masing-masing individu untuk berkomunikasi atau menganalisis tugas jika anggota kawanan berada dalam jarak dekat. Biasanya perilaku ini akan dikombinasikan dengan perilaku *swarm* lainnya untuk mencapai tujuan yang ingin dicapai.

   Swarm dengan elemen N *quadrotor* dengan kondisi yang telah disebutkan sebelumnya, idealnya dikehendaki untuk berkumpul dalam satu titik tunggal terpusat atau di sekitar titik tunggal tersebut hingga mencapai steady state $t = \infty$ dinyatakan sebagai:

$$\lim_{t\to\infty} \| x_i(t) - x_j(t) \| \leq \varepsilon \tag{3.2}$$

   dimana ε merupakan jumlah maksimum *swarm*. Fungsi $X_i(t)$ dan $X_j(t)$ merupakan parameter masukan dimana $i = 1,2,3,...,N$ dan $j = 1,2,3,...,N$ seperti yang didefinisikan dalam [3].

2. Proses Pencarian Titik *Centroid*

Usaha mencari titik *centroid* dapat digambarkan dengan pemetaan $\Gamma^n \to \Gamma$ untuk setiap nilai $i \in \Gamma^n$ dimana $\sigma(i)$ adalah titik yang memiliki tiga rentang nilai. Umumnya $\sigma(i) > 0, \sigma(i), \sigma(i) < 0, atau \sigma(i) = 0$ adalah nilai yang mungkin untuk titik-titik ini. Titik tersebut dapat mewakili daerah yang dinginkan di pada area tersebut. $\sigma(i)$ digunakan untuk mewakili suatu titik atau area yang diinginkan. $\sigma(i) > 0$ digunakan untuk menentukan area di mana *swarm* harus menghindari terjadinya tabrakan dan akan menghasilkan probabilitas bahwa *quadrotor* tidak akan melintasi titik ini. Di sisi lain $\sigma(i) < 0$ adalah titik yang mempunyai kemungkinan besar bahwa *swarm* akan melewati arah ini. Demikian juga jika $\sigma(i) = 0$ terdapat titik netral. Area yang lebih menguntungkan adalah nilai yang lebih rendah dan area yang kurang diminati seharusnya memiliki nilai lebih besar; Terakhir titik netral harus memiliki nilai 0.

3. Formasi Penerbangan

Sebuah manifestasi menggabungkan agregasi dan proses pencarian titik *centroid* merupakan proses formasi terbang. Hal utama untuk menciptakan formasi penerbangan yang stabil adalah jarak antara elemen *swarm* seharusnya dipertahankan saat bergerak. Diberikan jarak yang diinginkan $\{d_{ij} | i, j \in \{1,2,3,...,N\}, i \neq j\}$ formasi terbang dapat tercapai. Keadaan *steady state* digambarkan sebagai berikut:

$$\lim_{t \to \infty} \left| \|x_T(t) - x_j(t)\| - d_{ij} \right| \leq \varepsilon \text{ dengan } \forall i \neq j \in \{1,2,3,...,N\} \qquad (3.3)$$

4. Tracking quadrotor swarm

Koordinasi antar *quadrotor swarm* selanjutnya dapat diketahui posisinya yang disebut *tracking*. Untuk kasus *quadrotor swarm* pada umumnya perpindahan dari *quadrotor swarm* dari titik ke titik lainnya menuju *centroid* berkerja bersamaan dengan optimasinya atau apabila menggunakan formasi tertentu harus tetap terjaga. *Quadrotor swarm* dapat di

ketahui posisi trackingnya sebagai titik *centroid* kawanan pada waktu tertentu *t*.

$$\bar{x}(t) = \frac{1}{N}\sum_{i=1}^{N} x_i(t) \tag{3.4}$$

5. Karakteristik Sosial dalam *Swarm*

   Iterasi yang terjadi sudah melebihi batas waktu yang ditentukan Batas waktu ini menjadi hal yang penting dalam kondisi ini. Jika batas waktu terlalu singkat, pencarian akan berhenti sebelum solusi ditemukan. Jika batas waktu terlalu panjang, pada pencarian yang gagal akan terus berjalan walaupun *ANNSOM* sudah mencapai titik konvergen dimana pergerakan menjadi tidak signifikan.

6. Solusi yang bisa diterima sudah ditemukan

   Jika dalam pencarian ternyata ditemukan solusi yang dapat diterima, maka pencarian akan dihentikan. Batas diterima disini adalah ambang batas dimana solusi yang ditemukan sudah sesuai dengan apa yang diinginkan. Jika ambang batas terlalu jauh, solusi yang ditemukan mungkin tidak cukup sesuai dengan apa yang diinginkan. Sementara ambang batas yang terlalu kecil akan membuat *ANNSOM* cukup kesulitan untuk menemukan solusi yang dapat memenuhinya.

7. Tidak ada peningkatan dalam beberapa iterasi

   Terdapat beberapa cara untuk mengetahui apakah peningkatan masih cukup signifikan untuk dapat dilanjutkan atau tidak. Jika perubahan posisi partikel cukup kecil, maka dapat dipastikan *ANNSOM* sudah menuju ke titik konvergen. Cara lain, jika perhitungan kecepatan pada *ANNSOM* memberikan hasil yang mendekati nol maka pertikel hampir tidak bergerak, dan pembelajaran pada *ANNSOM* dapat dihentikan.

8. Radius normal dari kumpulan partikel mendekati nol

   Radius normal dihitung sebagai berikut :

$$R_{norm} = \frac{R_{\max}}{diameter(S)} \tag{3.5}$$

Diameter (S) merupakan diameter dari kumpulan partikel awal dan R_{max} merupakan jarak maksimum partikel dari *global best*.

$$R_{\max} = \left\| x_m - p_g \right\| m = 1,.....,n(partikel) \quad (3.6)$$

$$R_{\max} \geq \left\| x_i - p_g \right\| \forall = 1,.....,n(partikel) \quad (3.7)$$

Jika $R_{\max}$ mendekati nol, maka kumpulan partikel memiliki kemungkinan kecil untuk melakukan peningkatan. Perlu didefinisikan batasan cukup dekat dengan nol. Jika batasan terlalu besar, maka *ANNSOM* akan berhenti sebelum solusi optimum ditemukan. Sebaliknya, jika batasan terlalu kecil, akan dibutuhkan iterasi yang lebih panjang untuk dapat mencapai batasan tersebut.

9. Ketika perubahan solusi yang ditemukan mendekati nol

Kondisi ini terjadi dengan melihat perubahan solusi yang telah ditemukan. Perubahan solusi dihitung dengan persamaan 3.8.

$$f(x) = \frac{f(p_g(t) - f(p_g(t-1))}{f(p_g(t))} \quad (3.8)$$

$f(x)$ adalah *fitness function* yang menghitung seberapa besar solusi telah ditemukan pada posisi *x*. Jika *f* terlalu kecil, maka kumpulan partikel dapat diasumsikan sudah konvergen menuju satu titik. Di sini juga dibutuhkan batasan seberapa dekat perubahan yang terjadi dengan nol. Seperti pada kondisi-kondisi sebelumnya, jika ambang batas terlalu besar, maka *ANNSOM* akan berhenti sebelum solusi yang optimal ditemukan, begitu pula sebaliknya.

Algoritma *ANNSOM* bersifat konvergen dimana pada iterasi tertentu seluruh partikel akan menuju ke satu titik *global best*. Jika hal ini terjadi, maka kemungkinan pergerakan partikel-partikel tersebut tidak akan menjadi terlalu signifikan yang membuat perhitungan *ANNSOM* menjadi tidak efisien.

Berikut adalah langkah kerja dari distribusi *quadrotor* untuk mencari titik terdekat dengan *centroid*:

a. Pencarian *Centroid*

Tahap ini adalah tahap dimana agen mencari koordinat dari *centroid*. Pada tahap ini agen dapat bergerak bebas ataupun mengikuti algoritma tertentu. Agen harus bergerak ke seluruh daerah pencarian hingga menemukan posisi *centroid*. Pencarian secara individu dimungkinkan untuk dilakukan, tentunya dengan mengikuti algoritma tertentu. Tanpa menggunakan algoritma tertentu atau dengan kata lain menggunakan gerak *random*, maka lebih baik menggunakan metode lain untuk menjangkau seluruh ruang pencarian. Salah satu cara untuk mencapai hal tersebut adalah dengan pencarian berkelompok. Pencarian secara berkelompok dengan cara berpencar akan mempercepat proses penemuan koordinat *centroid* yang dimaksud dalam tahap ini. Jika sudah ditemukan, maka agen akan masuk dalam fase berikutnya.

b. Deklarasi Posisi Titik *Centroid*

Tahap terakhir dan yang paling penting adalah mendeklarasikan dan memberikan tanda terhadap posisi titik *centroid* yang telah ditemukan. Deklarasi posisi titik *centroid* dilakukan oleh anggota *swarm quadrotor* yang posisinya terdekat dengan titik *centroid*, selanjutnya *quadrotor* ini sebagai acuan dari agen dan melakukan komputasi koordinat dengan anggota swarm *quadrotor* untuk bergerak menuju *centroid.*

3.3 *Time Constant Delay*

Time constant delay biasanya dilambangkan dengan huruf Yunani τ (tau), adalah parameter yang menandai respons terhadap sebuah orde pertama *input step*, sistem orde-invariant linier (LTI). Konstanta waktu adalah unit karakteristik utama dari sistem LTI orde pertama.

Dalam domain waktu, pilihan yang umum digunakan untuk mengeksplorasi respons waktu adalah melalui *step response* menuju ke *step input*, atau respons impuls terhadap masukan fungsi Dirac delta. Dalam domain frekuensi

(misalnya, melihat transformasi Fourier dari step response, atau menggunakan input yang merupakan fungsi waktu sinusoidal sederhana), konstanta waktu juga menentukan *bandwidth* dari *time-invariant system* orde pertama, yaitu frekuensi di mana daya sinyal output turun sampai setengah dari nilai yang dimilikinya pada frekuensi rendah.

Konstanta waktu juga digunakan untuk mengkarakterisasi respons frekuensi dari berbagai sistem pemrosesan sinyal - pita magnetik, pemancar dan penerima radio, peralatan perekam dan pemuatan rekaman, dan filter digital - yang dapat dimodelkan atau diperkirakan oleh sistem LTI orde pertama. Contoh lainnya meliputi konstanta waktu yang digunakan dalam sistem kontrol untuk pengendali tindakan integral dan turunan, yang seringkali bersifat *pneumatic*, bukan listrik.

Konstanta waktu adalah fitur dari *lumped system analysis* (metode analisis kapasitas yang disatukan) untuk sistem termal, yang digunakan saat benda sejuk atau hangat secara seragam berada di bawah pengaruh pendinginan atau pemanasan konvektif.

Secara fisik, konstanta waktu mewakili waktu yang telah berlalu yang diperlukan agar respon sistem menuju ke nol jika sistem terus stabil pada tingkat awal, karena perubahan progresif dalam laju nol, maka kali ini respons tersebut akan benar-benar menurun nilainya menjadi $1/e \approx 36.8\%$ (dari *step decrease*). Dalam sistem yang semakin meningkat, konstanta waktu adalah waktu bagi *step response* sistem untuk mencapai $1-1/e \approx 63.2\%$ dari nilai (*asymptotic*) akhir (dari *step increase*). Dalam peluruhan radioaktif, konstanta waktu disebut sebagai konstanta peluruhan (λ), dan ini mewakili rata-rata masa pakai sistem peluruhan (seperti atom) sebelum meluruh, atau waktu yang dibutuhkan untuk segala sesuatunya kecuali peluruhan dari 36,8% atom. Untuk alasan ini, konstanta waktu lebih lama dari waktu paruh, yang merupakan waktu hanya untuk peluruhan dari 50% atom.

Sistem LTI orde pertama dicirikan oleh persamaan diferensial yang dinyatakan dalam (3.9).

$$\tau \frac{dV}{dt} + V = f(t) \tag{3.9}$$

dimana τ mewakili konstanta peluruhan eksponensial dan *V* adalah fungsi waktu *t*. Sisi ruas kanan merupakan fungsi *forcing functioni f* (*t*) yang menggambarkan fungsi penggerak eksternal waktu, yang dapat dianggap sebagai *input* sistem, dimana *V* (*t*) adalah respon, atau *output* sistem.

Dalam potensial aksi (atau bahkan dalam penyebaran sinyal pasif) di neuron, konstanta waktu τ dinyatakan dalam Persamaan 3.12.

$$\tau = r_m c_m \tag{3.12}$$

dengan r_m adalah resistensi di seluruh membran dan c_m adalah kapasitansi membran. Resistansi di seluruh membran adalah fungsi dari jumlah saluran ion terbuka dan kapasitansi adalah fungsi dari sifat bilayer lipid. Konstanta waktu digunakan untuk menggambarkan kenaikan dan penurunan tegangan membran, dimana kenaikan tersebut digambarkan dalam Persamaan 3.13.

$$V(t) = V\max(1 - e^{-t/\tau}) \tag{3.13}$$

dan penurunannya digambarkan dalam Persamaan 3.14.

$$V(t) = V\max e^{-t/\tau} \tag{3.14}$$

dengan satuan voltase dalam milivolt, waktu dalam hitungan detik, dan τ dalam hitungan detik. $V\max$ didefinisikan sebagai tegangan maksimum yang dicapai pada potensial aksi, di mana $V\max = r_m I$.

dimana r_m adalah resistensi di seluruh membran dan *I* adalah arus. Pengaturan untuk $t = \tau$ untuk kenaikan set *V*(*t*) sama dengan $0.63V_{\max}$. Ini berarti bahwa konstanta waktu adalah waktu yang berlalu setelah 63% dari $V_{\max}$ dicapai. Pengaturan untuk $t = \tau$ untuk penurunan set *V*(*t*) sama dengan to $0.37V_{\max}$, yang berarti bahwa konstanta waktu adalah waktu yang berlalu setelah turun menjadi 37% dari V_{maks}. Konstanta waktu yang lebih besar berarti, semakin lambat naik atau turunnya potensi neuron. Konstanta waktu yang lama dapat menghasilkan penjumlahan temporal, atau penjumlahan aljabar dari potensi berulang. Konstanta waktu yang singkat justru menghasilkan *coincidence detector* melalui penjumlahan spasial.

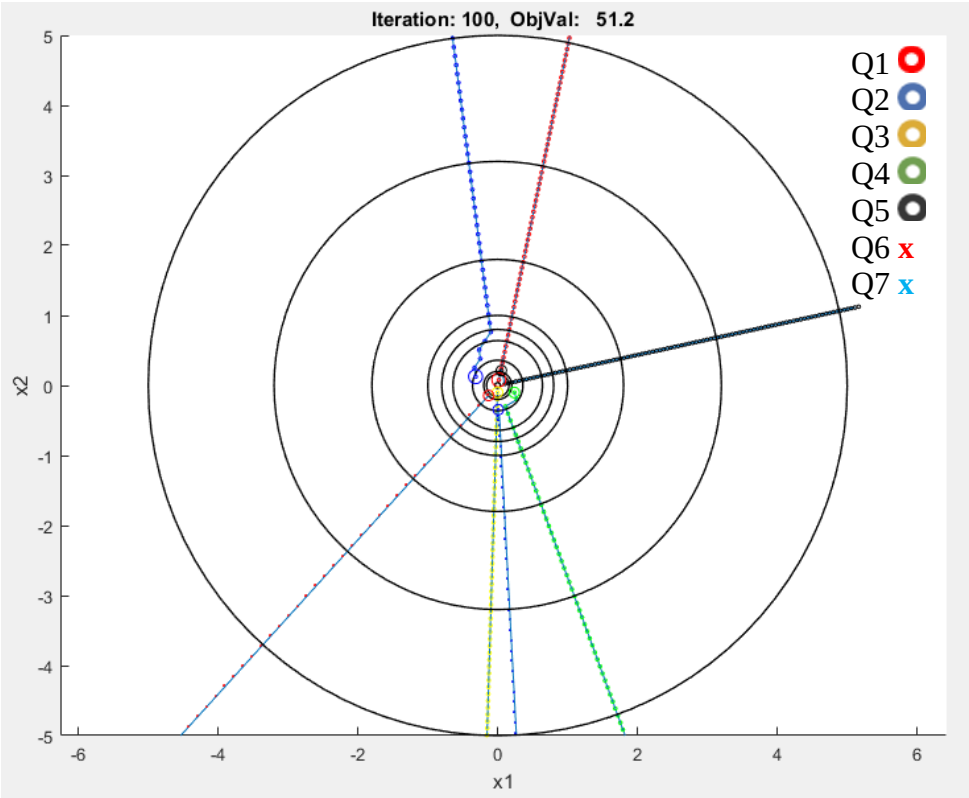
BAB 4
HASIL DAN PEMBAHASAN

Pengujian dilakukan dengan mengambil data terkait koordinat posisi masing-masing *quadrotor* sebagai anggota *swarm* dalam proses pelacakan *centroid* dengan asumsi bahwa pergerakan *quadrotor* pada sumbu *z* dengan ketinggian yang tetap, sehingga pergerakan *quadrotor swarm* direpresentasikan pada bidang koordinat *x* dan *y*, kemampuan *quadrotor* untuk melacak posisi titik *centroid* sehingga berada pada posisi terdekat dengan titik *centroid*, jarak antar anggota *quadrotor swarm* dalam proses pelacakan titik *centroid* dan jarak antar anggota *quadrotor swarm* sehingga mampu menghindari tabrakan antar anggota *quadrotor swarm*. Pengambilan data pengujian dilakukan sebanyak sepuluh kali, sehingga dapat dilakukan validitas data pengujian melalui uji varian pada data posisi pelacakan terdekat dengan titik *centroid* dan data jarak antar anggota *quadrotor swarm* untuk menghindari terjadinya tabrakan. Selain itu, dilakukan perbandingan data yang diperoleh saat menggunakan metode *modified ANNSOM* dengan metode *ANNSOM* pada paper [2].

4.1 Koordinat Posisi Masing-masing Anggota *Quadrotor Swarm* dalam Proses Pelacakan *Centroid*

Pada proses pelacakan titik *centroid* oleh anggota *quadrotor swarm*, posisi awal masing-masing *quadrotor* adalah acak. Dalam pengujian ini dilakukan pengambilan data sebanyak sepuluh kali untuk mengetahui posisi optimal masing-masing anggota *quadrotor swarm* dalam proses pelacakan titik *centroid*.

Pergerakan anggota *quadrotor swarm* dalam proses pelacakan titik *centroid* ditunjukkan pada Gambar 4.1. Sedangkan data koordinat posisi masing-masing anggota *swarm quadrotor* ditunjukkan pada Tabel 4.1. Jumlah anggota *quadrotor swarm* sebanyak tujuh dan posisi titik *centroid* yang akan dilacak pada koordinat (0,0).



Gambar 4. 1 Proses Pelacakan *Centroid* oleh Anggota *Quadrotor Swarm* pada Pengujian Pertama

Tabel 4. 1 Koordinat Posisi Anggota *Quadrotor Swarm* Pertama Hingga Keempat pada Sumbu *x* dan *y* saat Pengujian Pertama

Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y							
	Koordinat Q1		Koordinat Q2		Koordinat Q3		Koordinat Q4	
	x	*y*	*x*	*y*	*x*	*y*	*x*	*y*
1	1.65	8.00	-1.65	12.71	-0.30	-9.79	3.66	-10.03
2	1.64	7.92	-1.64	12.58	-0.30	-9.69	3.62	-9.93
3	1.62	7.84	-1.62	12.46	-0.29	-9.59	3.58	-9.83
4	1.60	7.76	-1.60	12.33	-0.29	-9.50	3.55	-9.73
5	1.59	7.68	-1.59	12.20	-0.29	-9.40	3.51	-9.63
6	1.57	7.60	-1.57	12.07	-0.29	-9.30	3.47	-9.53
7	1.55	7.52	-1.55	11.95	-0.28	-9.20	3.44	-9.43
8	1.54	7.44	-1.54	11.82	-0.28	-9.10	3.40	-9.33
9	1.52	7.36	-1.52	11.69	-0.28	-9.01	3.36	-9.23
dst	…	…	…	…	…	…	…	…
100	0.02	0.08	-0.32	0.13	0.00	-0.10	0.24	-0.10

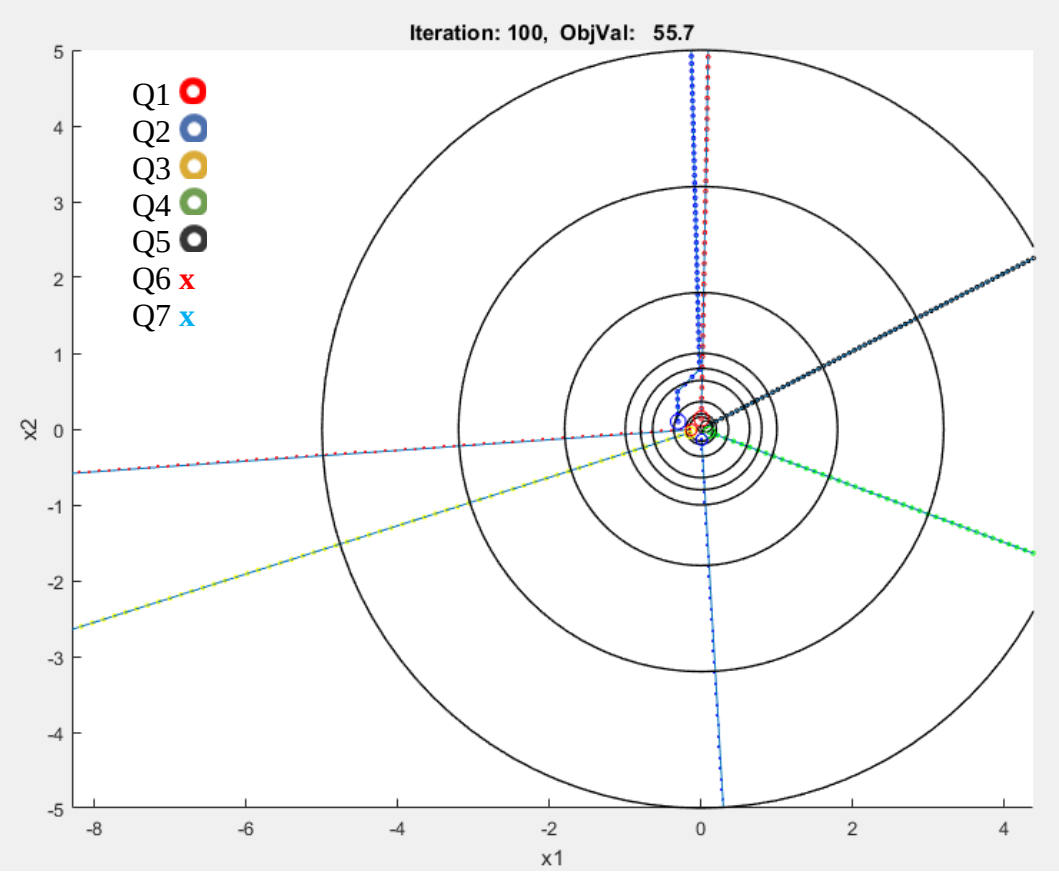
Tabel 4. 2 Koordinat Posisi Anggota *Quadrotor Swarm* Kelima Hingga Ketujuh pada Sumbu *x* dan *y* saat Pengujian Pertama

Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y					
	Koordinat Q5		Koordinat Q6		Koordinat Q7	
	x	*y*	*x*	*y*	*x*	*y*
1	5.16	1.12	-13.00	-14.33	0.78	-14.60
2	5.11	1.11	-12.87	-14.19	0.77	-14.45
3	5.06	1.10	-12.74	-14.04	0.77	-14.30
4	5.01	1.09	-12.61	-13.90	0.76	-14.16
5	4.96	1.08	-12.48	-13.76	0.75	-14.01
6	4.91	1.07	-12.35	-13.61	0.74	-13.87
7	4.85	1.06	-12.22	-13.47	0.73	-13.72
8	4.80	1.05	-12.09	-13.33	0.73	-13.57
9	4.75	1.03	-11.96	-13.18	0.72	-13.43
dst	…	…	…	…	…	…
100	0.05	0.21	-0.13	-0.14	0.01	-0.35

Berdasarkan data pada Tabel 4.1 dan Tabel 4.2 dapat diketahui bahwa dalam proses pergerakannya untuk melacak posisi titik *centroid* diperlukan seratus iterasi, untuk data pengujian secara lengkap dari iterasi pertama hingga iterasi terakhir dicantumkan dalam Lampiran B.1. Dari hasil pengujian diketahui bahwa, masing-masing mampu melakukan pelacakan posisi titik *centroid* secara optimal. Posisi anggota *quadrotor swarm* pertama saat iterasi terakhir pada koordinat (0.02, 0.08), posisi anggota *quadrotor swarm* kedua saat iterasi terakhir pada koordinat (-0.32, 0.13), posisi anggota *quadrotor swarm* ketiga saat iterasi terakhir pada koordinat (0.00, -0.10), posisi anggota *quadrotor swarm* keempat saat iterasi terakhir pada koordinat (0.24, -0.10), posisi anggota *quadrotor swarm* kelima saat iterasi terakhir pada koordinat (0.05, 0.21), posisi anggota *quadrotor swarm* keenam saat iterasi terakhir pada koordinat (-0.13, -0.14) dan posisi anggota *quadrotor swarm* ketujuh saat iterasi terakhir pada koordinat (0.01, -0.35).

Pengujian kedua dilakukan untuk mengamati proses pelacakan titik *centroid* oleh anggota *quadrotor swarm* dan membuktikan bahwa inisial posisi awal

dari anggota *quadrotor swarm* bersifat acak. Gambar 4.2 menunjukkan proses tersebut.



Gambar 4. 2 Proses Pelacakan *Centroid* oleh Anggota *Quadrotor Swarm* pada Pengujian Kedua

Berdasarkan hasil pelacakan titik *centroid* yang ditunjukkan pada Gambar 4.2 dapat diketahui bahwa anggota *quadrotor swarm* membentuk formasi yang berbeda jika dibandingkan pada pengujian pertama. Sehingga diperoleh pembuktian bahwa inisial posisi awal dari anggota *quadrotor swarm* bersifat acak. Data pengujian yang diperoleh berupa koordinat posisi masing-masing anggota *quadrotor swarm* pada masing-masing iterasi ditunjukkan pada Tabel 4.3.

Tabel 4. 3 Koordinat Posisi Anggota *Quadrotor Swarm* Pertama Hingga Keempat pada Sumbu *x* dan *y* saat Pengujian Kedua

Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y							
	Koordinat Q1		Koordinat Q2		Koordinat Q3		Koordinat Q4	
	x	*y*	*x*	*y*	*x*	*y*	*x*	*y*
1	0.26	13.64	-0.26	9.84	-14.60	-4.65	10.68	-3.99
2	0.25	13.50	-0.25	9.74	-14.45	-4.60	10.57	-3.95
3	0.25	13.37	-0.25	9.64	-14.31	-4.56	10.47	-3.91
4	0.25	13.23	-0.25	9.54	-14.16	-4.51	10.36	-3.87
5	0.25	13.10	-0.25	9.45	-14.01	-4.46	10.25	-3.83
6	0.24	12.96	-0.24	9.35	-13.87	-4.42	10.15	-3.79
7	0.24	12.82	-0.24	9.25	-13.72	-4.37	10.04	-3.75
8	0.24	12.69	-0.24	9.15	-13.58	-4.32	9.93	-3.71
9	0.24	12.55	-0.24	9.05	-13.43	-4.28	9.83	-3.67
dst	…	…	…	…	…	…	…	…
100	0.00	0.14	-0.30	0.10	-0.15	-0.05	0.11	-0.04

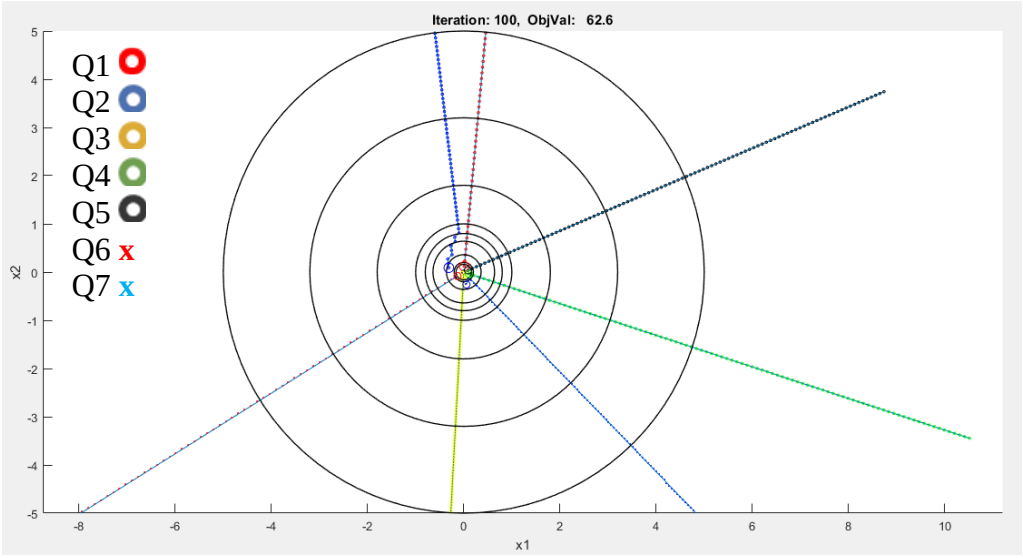
Tabel 4. 4 Koordinat Posisi Anggota *Quadrotor Swarm* Kelima Hingga Ketujuh pada Sumbu *x* dan *y* saat Pengujian Kedua

Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y					
	Koordinat Q5		Koordinat Q6		Koordinat Q7	
	x	*y*	*x*	*y*	*x*	*y*
1	6.74	3.47	-13.00	-0.92	0.82	-14.01
2	6.67	3.43	-12.87	-0.91	0.81	-13.87
3	6.61	3.40	-12.74	-0.90	0.81	-13.73
4	6.54	3.36	-12.61	-0.89	0.80	-13.59
5	6.47	3.33	-12.48	-0.88	0.79	-13.45
6	6.40	3.29	-12.35	-0.87	0.78	-13.31
7	6.34	3.26	-12.22	-0.86	0.77	-13.17
8	6.27	3.22	-12.09	-0.85	0.76	-13.03
9	6.20	3.19	-11.96	-0.84	0.76	-12.89
dst	…	…	…	…	…	…
100	0.07	0.03	-0.13	-0.01	0.01	-0.14

Dari hasil pengujian yang ditunjukkan pada Tabel 4.3 dan Tabel 4.4 diketahui bahwa, anggota *quadrotor swarm* yang mampu melakukan pelacakan terbaik dari posisi titik *centroid* (0,0) adalah anggota *quadrotor swarm* pertama dengan posisi koordinat (0.00, 0.14). Anggota *quadrotor swarm* pertama memiliki posisi terdekat dengan titik *centroid* jika dibandingkan dengan anggota yang lain yaitu, anggota Anggota *quadrotor swarm* kedua berada pada koordinat (-0.30,

0.10), Anggota *quadrotor swarm* ketiga berada pada koordinat (-0.15, -0.05), Anggota *quadrotor swarm* keempat berada pada koordinat (0.11, -0.04), Anggota *quadrotor swarm* kelima berada pada koordinat (0.07, 0.03), Anggota *quadrotor swarm* keenam berada pada koordinat (-0.13, -0.01) dan Anggota *quadrotor swarm* ketujuh berada pada koordinat (0.01, -0.14).

Tabel 4.3 dan Tabel 4.4 menunjukkan posisi koordinat dari masing-masing anggota *quadrotor swarm* dalam proses pelacakan titik *centroid*. Data yang ditunjukkan menunjukkan perubahan posisi pada setiap iterasi. Iterasi yang diperlukan sebanyak seratus iterasi. Data secara detail untuk pergerakan anggota *quadrotor swarm* dalam proses pelacakan titik *centroid* dari iterasi pertama hingga iterasi ke-seratus di tunjukkan pada Lampiran B.2. Pada kondisi ini anggota *quadrotor swarm* dinyatakan telah berada pada posisi optimal dalam hal melakukan pelacakan posisi titik *centroid.*



Gambar 4. 3 Proses Pelacakan *Centroid* oleh Anggota *Quadrotor Swarm* pada Pengujian Ketiga

Proses pelacakan titik *centroid* oleh anggota *quadrotor swarm* pada pengujian ketiga ditunjukkan pada Gambar 4.3 Proses pelacakan oleh masing-masing anggota *quadrotor swarm* dimulai dengan inisialisasi posisi awal yang acak dan posisi akhir yang terdekat dengan titik *centroid*. Pada proses pelacakan titik *centroid*, anggota *quadrotor swarm* kedua melakukan proses pembelokan arah,

karena jalur awal yang dilalui akan bertabrakan dengan anggota *quadrotor swarm* pertama.

Tabel 4. 5 Koordinat Posisi Anggota *Quadrotor Swarm* Pertama Hingga Keempat pada Sumbu *x* dan *y* saat Pengujian Ketiga

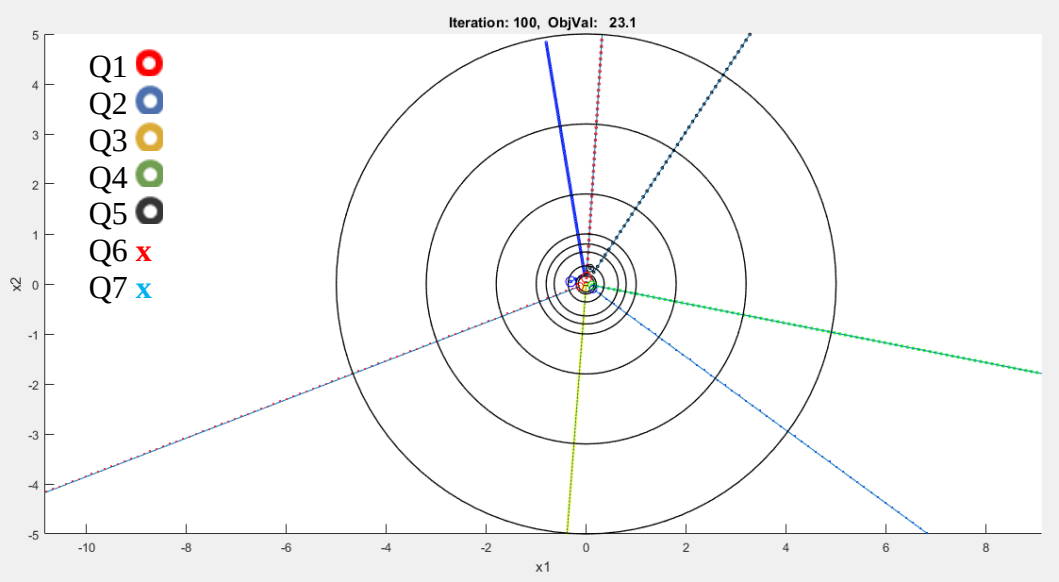
Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y							
	Koordinat Q1		Koordinat Q2		Koordinat Q3		Koordinat Q4	
	x	*y*	*x*	*y*	*x*	*y*	*x*	*y*
1	1.08	11.81	-1.08	9.00	-0.30	-5.56	10.52	-3.45
2	1.07	11.69	-1.07	8.91	-0.30	-5.51	10.41	-3.41
3	1.06	11.57	-1.06	8.82	-0.29	-5.45	10.31	-3.38
4	1.05	11.45	-1.05	8.73	-0.29	-5.39	10.20	-3.34
5	1.04	11.33	-1.04	8.64	-0.29	-5.34	10.10	-3.31
6	1.03	11.22	-1.03	8.55	-0.29	-5.28	9.99	-3.27
7	1.02	11.10	-1.02	8.46	-0.28	-5.23	9.89	-3.24
8	1.01	10.98	-1.01	8.37	-0.28	-5.17	9.78	-3.21
9	1.00	10.86	-1.00	8.28	-0.28	-5.12	9.68	-3.17
dst	…	…	…	…	…	…	…	…
100	0.01	0.12	-0.31	0.09	0.00	-0.06	0.11	-0.03

Tabel 4. 6 Koordinat Posisi Anggota *Quadrotor Swarm* Kelima Hingga Ketujuh pada Sumbu *x* dan *y* saat Pengujian Ketiga

Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y					
	Koordinat Q5		Koordinat Q6		Koordinat Q7	
	x	*y*	*x*	*y*	*x*	*y*
1	6.74	3.47	-13.00	-0.92	0.82	-14.01
2	6.67	3.43	-12.87	-0.91	0.81	-13.87
3	6.61	3.40	-12.74	-0.90	0.81	-13.73
4	6.54	3.36	-12.61	-0.89	0.80	-13.59
5	6.47	3.33	-12.48	-0.88	0.79	-13.45
6	6.40	3.29	-12.35	-0.87	0.78	-13.31
7	6.34	3.26	-12.22	-0.86	0.77	-13.17
8	6.27	3.22	-12.09	-0.85	0.76	-13.03
9	6.20	3.19	-11.96	-0.84	0.76	-12.89
dst	…	…	…	…	…	…
100	0.07	0.03	-0.13	-0.01	0.01	-0.14

Berdasarkan Tabel 4.5 dan Tabel 4.6 diketahui bahwa inisial posisi awal terjauh dimulai oleh anggota *quadrotor swarm* keenam yaitu pada posisi awal (-13.00, -0.92). Proses pelacakan berhenti pada iterasi ke-seratus karena anggota *swarm quadrotor* telah menemukan solusi optimalnya, dimana posisi ini

menyatakan koordinat yang terdekat dengan koordinat titik *centroid*. Data koordinat posisi setiap iterasi secara lengkap tercantum pada Lampiran B.3.



Gambar 4. 4 Proses Pelacakan *Centroid* oleh Anggota *Quadrotor Swarm* pada Pengujian Keempat

Tabel 4. 7 Koordinat Posisi Anggota *Quadrotor Swarm* Pertama Hingga Keempat pada Sumbu *x* dan *y* saat Pengujian Keempat

Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y							
	Koordinat Q1		Koordinat Q2		Koordinat Q3		Koordinat Q4	
	x	*y*	*x*	*y*	*x*	*y*	*x*	*y*
1	0.80	12.50	-0.80	4.83	-0.53	-6.92	11.28	-2.22
2	0.79	12.38	-0.79	4.78	-0.52	-6.85	11.17	-2.20
3	0.78	12.25	-0.78	4.73	-0.52	-6.78	11.05	-2.17
4	0.78	12.13	-0.78	4.68	-0.51	-6.72	10.94	-2.15
5	0.77	12.00	-0.77	4.64	-0.51	-6.65	10.83	-2.13
6	0.76	11.88	-0.76	4.59	-0.50	-6.58	10.71	-2.11
7	0.75	11.75	-0.75	4.54	-0.50	-6.51	10.60	-2.08
8	0.74	11.63	-0.74	4.49	-0.49	-6.44	10.49	-2.06
9	0.74	11.50	-0.74	4.44	-0.49	-6.37	10.38	-2.04
dst	…	…	…	…	…	…	…	…
100	0.01	0.13	-0.31	0.05	-0.01	-0.07	0.11	-0.02

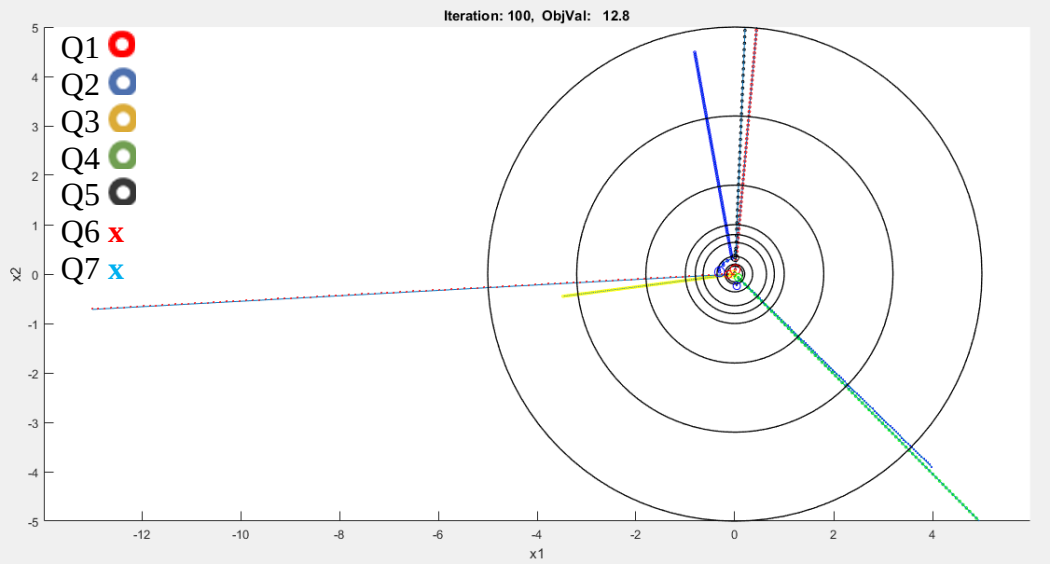
Berdasarkan data pada Tabel 4.7 dapat diketahui bahwa dalam proses pergerakannya untuk melacak posisi titik *centroid* diperlukan seratus iterasi. Dari hasil pengujian diketahui bahwa, masing-masing mampu melakukan pelacakan posisi titik *centroid* secara optimal. Posisi anggota *quadrotor swarm* pertama saat

iterasi terakhir pada koordinat (0.01, 0.13), posisi anggota *quadrotor swarm* kedua saat iterasi terakhir pada koordinat (-0.31, 0.05), posisi anggota *quadrotor swarm* ketiga saat iterasi terakhir pada koordinat (-0.01, -0.07), dan posisi anggota *quadrotor swarm* keempat saat iterasi terakhir pada koordinat (0.11, -0.02). Sedangkan koordinat posisi anggota *quadrotor swarm* kelima hingga ketujuh pada sumbu *x* dan *y* saat pengujian keempat ditunjukkan pada Tabel 4.8. Sedangkan data koordinat posisi secara lengkap tercantum dalam Lampiran B.4.

Tabel 4. 8 Koordinat Posisi Anggota *Quadrotor Swarm* Kelima Hingga Ketujuh pada Sumbu *x* dan *y* saat Pengujian Keempat

Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y					
	Koordinat Q5		Koordinat Q6		Koordinat Q7	
	x	*y*	*x*	*y*	*x*	*y*
1	7.61	11.61	-13.00	-5.01	13.93	-10.19
2	7.54	11.49	-12.87	-4.96	13.79	-10.09
3	7.46	11.38	-12.74	-4.91	13.65	-9.98
4	7.38	11.26	-12.61	-4.86	13.51	-9.88
5	7.31	11.14	-12.48	-4.81	13.37	-9.78
6	7.23	11.03	-12.35	-4.76	13.23	-9.68
7	7.16	10.91	-12.22	-4.71	13.09	-9.58
8	7.08	10.80	-12.09	-4.66	12.95	-9.47
9	7.00	10.68	-11.96	-4.61	12.82	-9.37
dst	…	…	…	…	…	…
100	0.08	0.32	-0.13	-0.05	0.14	-0.10

Pengujian kelima dilakukan untuk mengamati proses pelacakan titik *centroid* oleh anggota *quadrotor swarm* ditunjukkan pada Gambar 4.5.



Gambar 4. 5 Proses Pelacakan *Centroid* oleh Anggota *Quadrotor Swarm* pada Pengujian Kelima

Tabel 4. 9 Koordinat Posisi Anggota *Quadrotor Swarm* Pertama Hingga Keempat pada Sumbu *x* dan *y* saat Pengujian Kelima

Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y							
	Koordinat Q1		Koordinat Q2		Koordinat Q3		Koordinat Q4	
	x	*y*	*x*	*y*	*x*	*y*	*x*	*y*
1	0.81	9.13	-0.81	4.49	-3.48	-0.45	6.79	-6.86
2	0.80	9.04	-0.80	4.44	-3.44	-0.45	6.72	-6.80
3	0.79	8.95	-0.79	4.40	-3.41	-0.44	6.65	-6.73
4	0.79	8.86	-0.79	4.35	-3.37	-0.44	6.59	-6.66
5	0.78	8.77	-0.78	4.31	-3.34	-0.43	6.52	-6.59
6	0.77	8.68	-0.77	4.26	-3.30	-0.43	6.45	-6.52
7	0.76	8.58	-0.76	4.22	-3.27	-0.42	6.38	-6.45
8	0.75	8.49	-0.75	4.17	-3.23	-0.42	6.31	-6.38
9	0.75	8.40	-0.75	4.13	-3.20	-0.41	6.25	-6.31
dst	…	…	…	…	…	…	…	…
100	0.01	0.09	-0.31	0.04	-0.03	0.00	0.07	-0.07

Dari hasil pengujian yang ditunjukkan pada Tabel 4.9 dan Tabel 4.10 diketahui bahwa, anggota *quadrotor swarm* yang mampu melakukan pelacakan terbaik dari posisi titik *centroid* (0,0) adalah anggota *quadrotor swarm* ketiga dengan posisi koordinat (-0.03, 0.00). Anggota *quadrotor swarm* ketiga memiliki

posisi terdekat dengan titik *centroid* jika dibandingkan dengan anggota yang lain yaitu, anggota *quadrotor swarm* pertama berada pada koordinat (0.01, 0.09), anggota *quadrotor swarm* kedua berada pada koordinat (-0.31, 0.04), anggota *quadrotor swarm* keempat berada pada koordinat (0.07, -0.07), anggota *quadrotor swarm* kelima berada pada koordinat (0.01, 0.33), anggota *quadrotor swarm* keenam berada pada koordinat (-0.13, -0.01), dan anggota *quadrotor swarm* ketujuh berada pada koordinat (0.04, -0.24). Data secara detail tercantum dalam Lampiran B.5.

Tabel 4. 10 Koordinat Posisi Anggota *Quadrotor Swarm* Kelima Hingga Ketujuh pada Sumbu *x* dan *y* saat Pengujian Kelima

Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y					
	Koordinat Q5		Koordinat Q6		Koordinat Q7	
	x	*y*	*x*	*y*	*x*	*y*
1	0.54	12.98	-13.00	-0.72	3.99	-3.91
2	0.54	12.85	-12.87	-0.71	3.95	-3.87
3	0.53	12.72	-12.74	-0.70	3.91	-3.83
4	0.53	12.59	-12.61	-0.69	3.87	-3.79
5	0.52	12.46	-12.48	-0.69	3.83	-3.75
6	0.51	12.33	-12.35	-0.68	3.79	-3.71
7	0.51	12.20	-12.22	-0.67	3.75	-3.67
8	0.50	12.07	-12.09	-0.67	3.71	-3.63
9	0.50	11.94	-11.96	-0.66	3.67	-3.59
dst	…	…	…	…	…	…
100	0.01	0.33	-0.13	-0.01	0.04	-0.24

Tabel 4.11 dan Tabel 4.12 menunjukkan posisi koordinat dari masing-masing anggota *quadrotor swarm* dalam proses pelacakan titik *centroid*. Data yang ditunjukkan menunjukkan perubahan posisi pada setiap iterasi secara detail tercantum dalam Lampiran B.6. Iterasi yang diperlukan sebanyak seratus iterasi. Pada kondisi ini anggota *quadrotor swarm* dinyatakan telah berada pada posisi optimal dalam hal melakukan pelacakan posisi titik *centroid.*

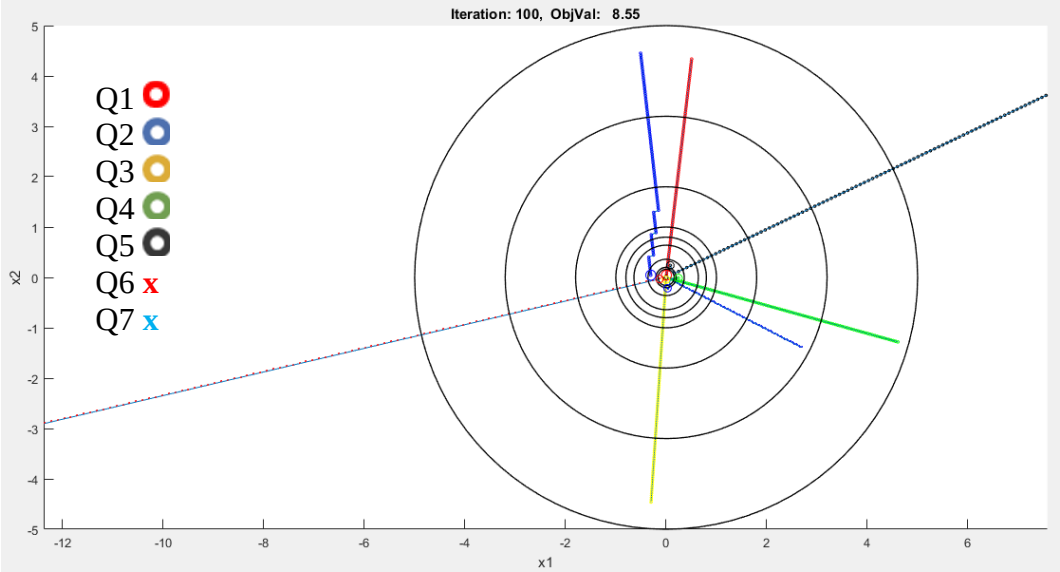
Tabel 4. 11 Koordinat Posisi Anggota *Quadrotor Swarm* Pertama Hingga Keempat pada Sumbu *x* dan *y* saat Pengujian Keenam

Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y							
	Koordinat Q1		Koordinat Q2		Koordinat Q3		Koordinat Q4	
	x	*y*	*x*	*y*	*x*	*y*	*x*	*y*
1	0.51	4.34	-0.51	4.45	-0.30	-4.46	4.61	-1.28
2	0.50	4.29	-0.50	4.41	-0.30	-4.42	4.56	-1.27
3	0.50	4.25	-0.50	4.36	-0.29	-4.37	4.52	-1.26
4	0.49	4.21	-0.49	4.32	-0.29	-4.33	4.47	-1.24
5	0.49	4.16	-0.49	4.27	-0.29	-4.28	4.42	-1.23
6	0.48	4.12	-0.48	4.23	-0.29	-4.24	4.38	-1.22
7	0.48	4.08	-0.48	4.18	-0.28	-4.19	4.33	-1.20
8	0.47	4.03	-0.47	4.14	-0.28	-4.15	4.29	-1.19
9	0.47	3.99	-0.47	4.09	-0.28	-4.11	4.24	-1.18
dst	…	…	…	…	…	…	…	…
100	0.01	0.04	-0.31	0.04	0.00	-0.04	0.25	-0.01

Tabel 4. 12 Koordinat Posisi Anggota *Quadrotor Swarm* Kelima Hingga Ketujuh pada Sumbu *x* dan *y* saat Pengujian Keenam

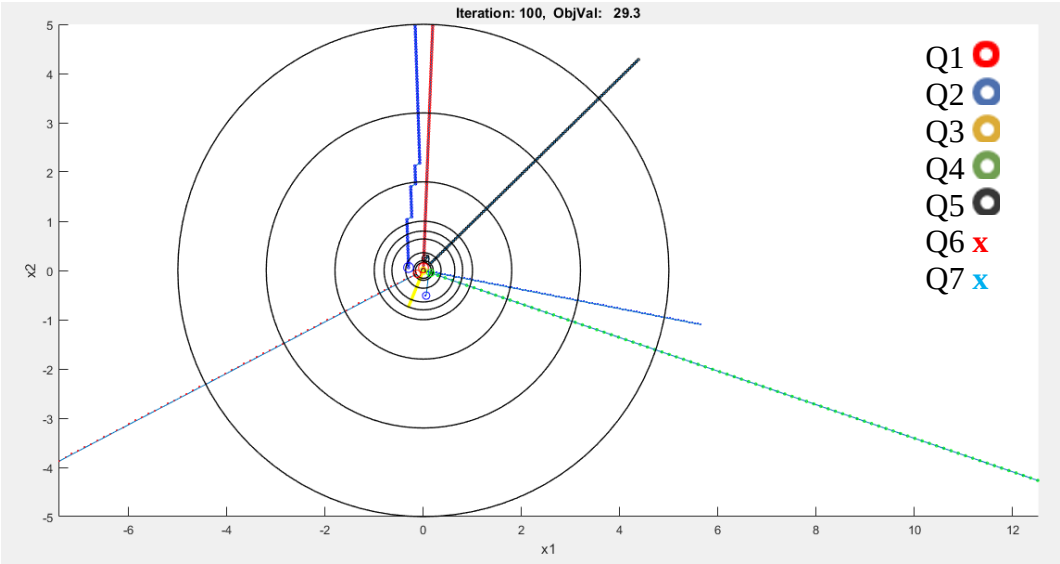
Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y					
	Koordinat Q5		Koordinat Q6		Koordinat Q7	
	x	*y*	*x*	*y*	*x*	*y*
1	8.21	3.93	-13.00	-3.05	2.70	-1.39
2	8.13	3.89	-12.87	-3.02	2.67	-1.38
3	8.05	3.85	-12.74	-2.99	2.65	-1.37
4	7.96	3.81	-12.61	-2.96	2.62	-1.35
5	7.88	3.77	-12.48	-2.93	2.59	-1.34
6	7.80	3.73	-12.35	-2.90	2.56	-1.32
7	7.72	3.69	-12.22	-2.87	2.54	-1.31
8	7.64	3.65	-12.09	-2.84	2.51	-1.30
9	7.55	3.61	-11.96	-2.81	2.48	-1.28
dst	…	…	…	…	…	…
100	0.08	0.24	-0.13	-0.03	0.03	-0.21

Proses pelacakan titik *centroid* oleh anggota *quadrotor swarm* pada pengujian keenam ditunjukkan pada Gambar 4.6. Proses pelacakan oleh masing-masing anggota *quadrotor swarm* dimulai dengan inisialisasi posisi awal yang acak dan posisi akhir yang terdekat dengan titik *centroid*.



Gambar 4. 6 Proses Pelacakan *Centroid* oleh Anggota *Quadrotor Swarm* pada Pengujian Keenam

Pergerakan anggota *quadrotor swarm* dalam proses pelacakan titik *centroid* pada pengujian ketujuh ditunjukkan dalam Gambar 4.7. Sedangkan data koordinat posisi masing-masing anggota *swarm quadrotor* ditunjukkan pada Tabel 4.13 dan Tabel 4.14, dan data secara detail tercantum dalam Lampiran B.7.



Gambar 4. 7 Proses Pelacakan *Centroid* oleh Anggota *Quadrotor Swarm* pada Pengujian Ketujuh

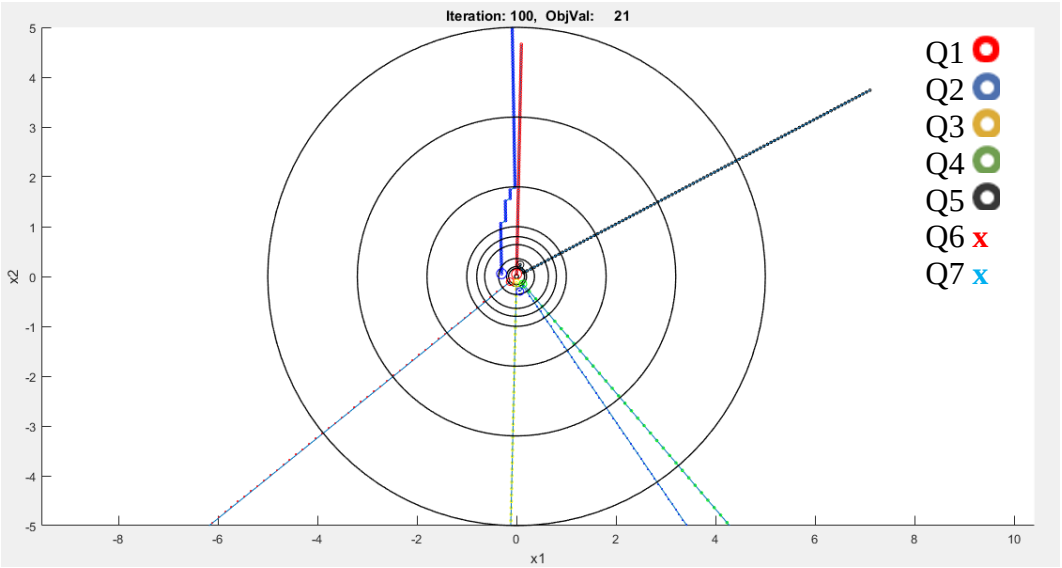
Tabel 4. 13 Koordinat Posisi Anggota *Quadrotor Swarm* Pertama Hingga Keempat pada Sumbu *x* dan *y* saat Pengujian Ketujuh

Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y							
	Koordinat Q1		Koordinat Q2		Koordinat Q3		Koordinat Q4	
	x	*y*	*x*	*y*	*x*	*y*	*x*	*y*
1	0.20	5.33	-0.20	6.06	-0.30	-0.73	14.72	-5.02
2	0.20	5.28	-0.20	6.00	-0.30	-0.73	14.57	-4.97
3	0.20	5.22	-0.20	5.94	-0.29	-0.72	14.43	-4.92
4	0.20	5.17	-0.20	5.88	-0.29	-0.71	14.28	-4.87
5	0.20	5.12	-0.20	5.82	-0.29	-0.70	14.13	-4.82
6	0.19	5.06	-0.19	5.76	-0.29	-0.70	13.98	-4.77
7	0.19	5.01	-0.19	5.70	-0.28	-0.69	13.84	-4.72
8	0.19	4.96	-0.19	5.64	-0.28	-0.68	13.69	-4.67
9	0.19	4.90	-0.19	5.58	-0.28	-0.67	13.54	-4.62
dst	…	…	…	…	…	…	…	…
100	0.00	0.05	-0.30	0.06	0.00	-0.01	0.15	-0.05

Tabel 4. 14 Koordinat Posisi Anggota *Quadrotor Swarm* Kelima Hingga Ketujuh pada Sumbu *x* dan *y* saat Pengujian Ketujuh

Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y					
	Koordinat Q5		Koordinat Q6		Koordinat Q7	
	x	*y*	*x*	*y*	*x*	*y*
1	4.38	4.28	-13.00	-6.79	5.66	-1.10
2	4.34	4.24	-12.87	-6.72	5.60	-1.09
3	4.29	4.20	-12.74	-6.65	5.55	-1.08
4	4.25	4.15	-12.61	-6.58	5.49	-1.07
5	4.20	4.11	-12.48	-6.52	5.43	-1.06
6	4.16	4.07	-12.35	-6.45	5.38	-1.05
7	4.12	4.03	-12.22	-6.38	5.32	-1.04
8	4.07	3.98	-12.09	-6.31	5.26	-1.02
9	4.03	3.94	-11.96	-6.24	5.21	-1.01
dst	…	…	…	…	…	…
100	0.04	0.24	-0.13	-0.07	0.06	-0.51

Proses pelacakan titik *centroid* oleh anggota *quadrotor* pada pengujian kedelapan ditunjukkan pada Gambar 4.8. Data koordinat posisi setiap iterasi secara detail tercantum dalam Lampiran B.8.



Gambar 4. 8 Proses Pelacakan *Centroid* oleh Anggota *Quadrotor Swarm* pada Pengujian Kedelapan

Tabel 4. 15 Koordinat Posisi Anggota *Quadrotor Swarm* Pertama Hingga Keempat pada Sumbu *x* dan *y* saat Pengujian Kedelapan

Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y							
	Koordinat Q1		Koordinat Q2		Koordinat Q3		Koordinat Q4	
	x	*y*	*x*	*y*	*x*	*y*	*x*	*y*
1	0.10	4.66	-0.10	5.60	-0.30	-12.90	12.81	-14.96
2	0.10	4.61	-0.10	5.54	-0.30	-12.77	12.69	-14.81
3	0.10	4.56	-0.10	5.49	-0.29	-12.64	12.56	-14.66
4	0.09	4.52	-0.09	5.43	-0.29	-12.51	12.43	-14.51
5	0.09	4.47	-0.09	5.37	-0.29	-12.38	12.30	-14.37
6	0.09	4.43	-0.09	5.32	-0.29	-12.25	12.17	-14.22
7	0.09	4.38	-0.09	5.26	-0.28	-12.12	12.05	-14.07
8	0.09	4.33	-0.09	5.21	-0.28	-12.00	11.92	-13.92
9	0.09	4.29	-0.09	5.15	-0.28	-11.87	11.79	-13.77
dst	…	…	…	…	…	…	…	…
100	0.00	0.05	-0.30	0.06	0.00	-0.13	0.13	-0.15

Tabel 4. 16 Koordinat Posisi Anggota *Quadrotor Swarm* Kelima Hingga Ketujuh pada Sumbu *x* dan *y* saat Pengujian Kedelapan

Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y					
	Koordinat Q5		Koordinat Q6		Koordinat Q7	
	x	*y*	*x*	*y*	*x*	*y*
1	7.10	3.74	-13.00	-10.52	6.97	-10.18
2	7.03	3.70	-12.87	-10.42	6.90	-10.08
3	6.96	3.66	-12.74	-10.31	6.83	-9.98
4	6.89	3.63	-12.61	-10.21	6.76	-9.88
5	6.82	3.59	-12.48	-10.10	6.69	-9.77
6	6.75	3.55	-12.35	-10.00	6.62	-9.67
7	6.68	3.51	-12.22	-9.89	6.55	-9.57
8	6.61	3.48	-12.09	-9.79	6.48	-9.47
9	6.53	3.44	-11.96	-9.68	6.41	-9.37
dst	…	…	…	…	…	…
100	0.07	0.24	-0.13	-0.11	0.07	-0.30

Tabel 4.17 dan Tabel 4.18 menunjukkan posisi koordinat dari masing-masing anggota *quadrotor swarm* dalam proses pelacakan titik *centroid* pada pengujian kesembilan. Pada kondisi iterasi terakhir, anggota *quadrotor swarm* dinyatakan telah berada pada posisi optimal dalam hal melakukan pelacakan posisi titik *centroid.* Data koordinat posisi setiap iterasi secara detail tercantum dalam Lampiran B.9.

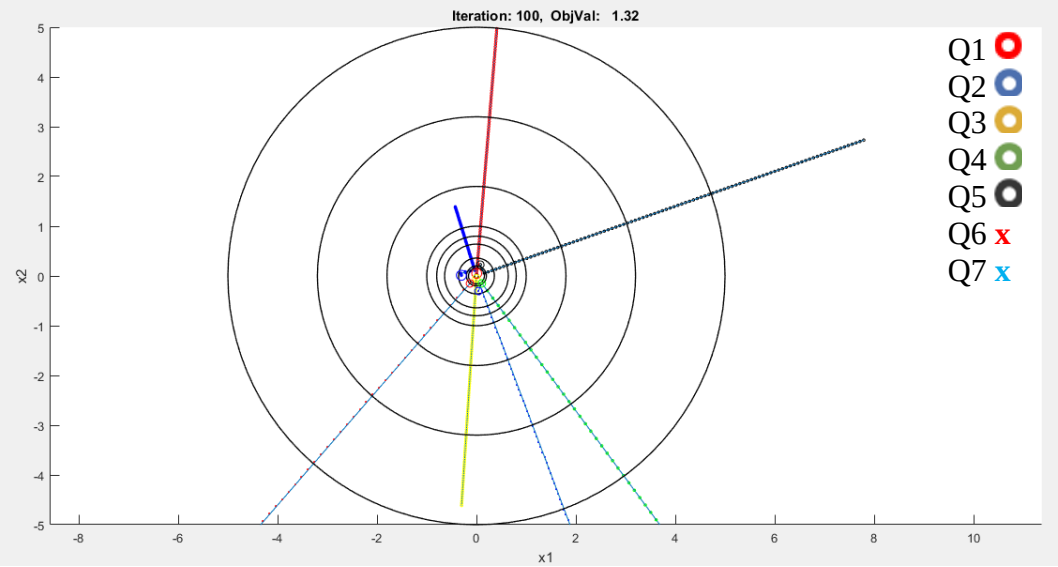
Tabel 4. 17 Koordinat Posisi Anggota *Quadrotor Swarm* Pertama Hingga Keempat pada Sumbu *x* dan *y* saat Pengujian Kesembilan

Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y							
	Koordinat Q1		Koordinat Q2		Koordinat Q3		Koordinat Q4	
	x	*y*	*x*	*y*	*x*	*y*	*x*	*y*
1	0.43	5.18	-0.43	1.39	-0.30	-4.62	10.95	-14.85
2	0.42	5.13	-0.42	1.38	-0.30	-4.57	10.84	-14.70
3	0.42	5.07	-0.42	1.36	-0.29	-4.52	10.73	-14.55
4	0.41	5.02	-0.41	1.35	-0.29	-4.48	10.62	-14.41
5	0.41	4.97	-0.41	1.34	-0.29	-4.43	10.51	-14.26
6	0.41	4.92	-0.41	1.32	-0.29	-4.38	10.40	-14.11
7	0.40	4.87	-0.40	1.31	-0.28	-4.34	10.30	-13.96
8	0.40	4.81	-0.40	1.29	-0.28	-4.29	10.19	-13.81
9	0.39	4.76	-0.39	1.28	-0.28	-4.25	10.08	-13.66
dst	…	…	…	…	…	…	…	…
100	0.00	0.05	-0.30	0.01	0.00	-0.05	0.11	-0.15

Tabel 4. 18 Koordinat Posisi Anggota *Quadrotor Swarm* Kelima Hingga Ketujuh pada Sumbu *x* dan *y* saat Pengujian Kesembilan

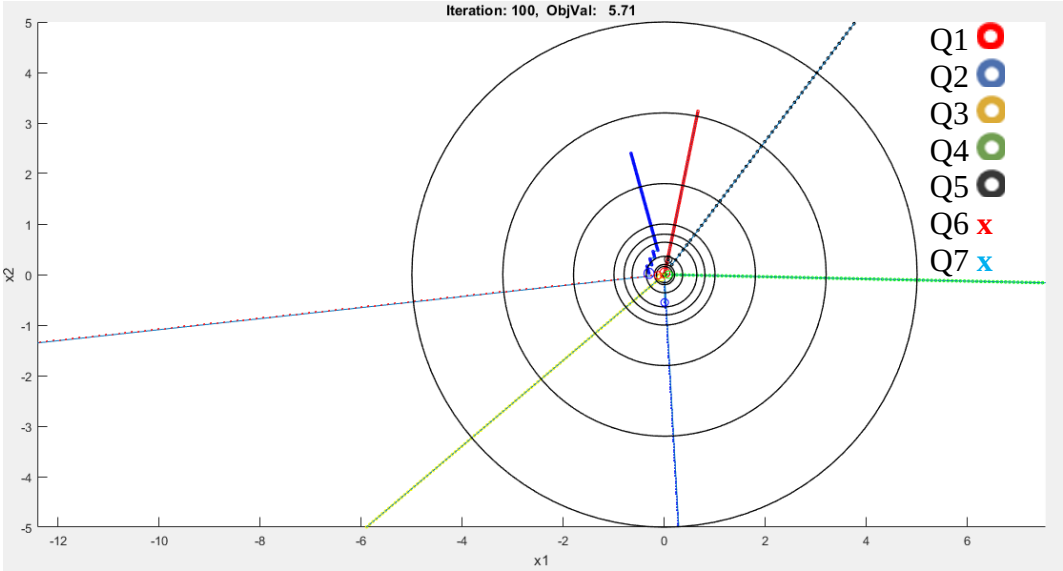
Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y					
	Koordinat Q5		Koordinat Q6		Koordinat Q7	
	x	*y*	*x*	*y*	*x*	*y*
1	7.79	2.73	-13.00	-14.97	3.93	-10.46
2	7.71	2.70	-12.87	-14.82	3.89	-10.35
3	7.64	2.68	-12.74	-14.67	3.86	-10.25
4	7.56	2.65	-12.61	-14.52	3.82	-10.14
5	7.48	2.62	-12.48	-14.37	3.78	-10.04
6	7.40	2.59	-12.35	-14.22	3.74	-9.93
7	7.32	2.57	-12.22	-14.07	3.70	-9.83
8	7.25	2.54	-12.09	-13.92	3.66	-9.73
9	7.17	2.51	-11.96	-13.77	3.62	-9.62
dst	…	…	…	…	…	…
100	0.08	0.23	-0.13	-0.15	0.04	-0.30

Proses pelacakan titik *centroid* oleh anggota *quadrotor swarm* pada pengujian kesembilan ditunjukkan pada Gambar 4.9. Proses pelacakan oleh masing-masing anggota *quadrotor swarm* dimulai dengan inisialisasi posisi awal yang acak dan posisi akhir yang terdekat dengan titik *centroid*. Jumlah iterasi yang diperlukan untuk proses pelacakan titik *centroid* sejumlah seratus iterasi.



Gambar 4. 9 Proses Pelacakan *Centroid* oleh Anggota *Quadrotor Swarm* pada Pengujian Kesembilan

Pengujian kesepuluh untuk mengamati pergerakan anggota *quadrotor swarm* dalam proses pelacakan titik *centroid* ditunjukkan pada Gambar 4.10. Sedangkan data koordinat posisi masing-masing anggota *swarm quadrotor* ditunjukkan pada Tabel 4.19 dan Tabel 4.20. Data koordinat posisi setiap iterasi secara detail tercantum dalam Lampiran B.10.



Gambar 4. 10 Proses Pelacakan *Centroid* oleh Anggota *Quadrotor Swarm* pada Pengujian Kesepuluh

Tabel 4. 19 Koordinat Posisi Anggota *Quadrotor Swarm* Pertama Hingga Keempat pada Sumbu *x* dan *y* saat Pengujian Kesepuluh

Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y							
	Koordinat Q1		Koordinat Q2		Koordinat Q3		Koordinat Q4	
	x	*y*	*x*	*y*	*x*	*y*	*x*	*y*
1	0.66	3.23	-0.66	2.40	-6.27	-5.32	8.13	-0.18
2	0.66	3.20	-0.66	2.38	-6.21	-5.26	8.05	-0.17
3	0.65	3.17	-0.65	2.35	-6.15	-5.21	7.97	-0.17
4	0.64	3.14	-0.64	2.33	-6.08	-5.16	7.89	-0.17
5	0.64	3.10	-0.64	2.30	-6.02	-5.10	7.81	-0.17
6	0.63	3.07	-0.63	2.28	-5.96	-5.05	7.73	-0.17
7	0.62	3.04	-0.62	2.26	-5.90	-5.00	7.65	-0.17
8	0.62	3.01	-0.62	2.23	-5.83	-4.94	7.57	-0.16
9	0.61	2.97	-0.61	2.21	-5.77	-4.89	7.48	-0.16
dst	…	…	…	…	…	…	…	…
100	0.01	0.03	-0.31	0.02	-0.06	-0.05	0.08	0.00

Tabel 4. 20 Koordinat Posisi Anggota *Quadrotor Swarm* Kelima Hingga Ketujuh pada Sumbu *x* dan *y* saat Pengujian Kesepuluh

Iterasi ke-	Koordinat posisi masing-masing anggota quadrotor swarm pada sumbu x dan y					
	Koordinat Q5		Koordinat Q6		Koordinat Q7	
	x	*y*	*x*	*y*	*x*	*y*
1	7.35	9.74	-13.00	-1.42	0.29	-5.19
2	7.28	9.64	-12.87	-1.40	0.29	-5.14
3	7.21	9.54	-12.74	-1.39	0.29	-5.08
4	7.13	9.45	-12.61	-1.38	0.28	-5.03
5	7.06	9.35	-12.48	-1.36	0.28	-4.98
6	6.99	9.25	-12.35	-1.35	0.28	-4.93
7	6.91	9.15	-12.22	-1.33	0.28	-4.88
8	6.84	9.06	-12.09	-1.32	0.27	-4.82
9	6.77	8.96	-11.96	-1.30	0.27	-4.77
dst	…	…	…	…	…	…
100	0.07	0.30	-0.13	-0.01	0.00	-0.55

Berdasarkan data pada Tabel 4.10 dapat diketahui bahwa masing-masing anggota *quadrotor swarm* mampu melakukan pelacakan posisi titik *centroid* secara optimal. Posisi anggota *quadrotor swarm* pertama saat iterasi terakhir pada koordinat (0.01, 0.03), posisi anggota *quadrotor swarm* kedua saat iterasi terakhir pada koordinat (-0.31, 0.02), posisi anggota *quadrotor swarm* ketiga saat iterasi terakhir pada koordinat (-0.06, -0.05), posisi anggota *quadrotor swarm* keempat saat iterasi terakhir pada koordinat (0.08, 0.00), posisi anggota *quadrotor swarm* kelima saat iterasi terakhir pada koordinat (0.07, 0.30), posisi anggota *quadrotor swarm* keenam saat iterasi terakhir pada koordinat (-0.13, -0.01) dan posisi anggota *quadrotor swarm* ketujuh saat iterasi terakhir pada koordinat (0.00, -0.55).

4.2 Nilai Varian Posisi Pelacakan Terdekat dengan Titik *Centroid*

Pada pengujian sebelumnya dilakukan pengambilan data koordinat posisi masing-masing anggota *quadrotor swarm* untuk mengetahui kemampuan *quadrotor swarm* dalam melakukan pelacakan titik *centroid.* Pada pengujian ini, dilakukan pengambilan data posisi anggota *quadrotor swarm* yang terdekat dengan titik *centroid* pada koordinat (0,0) pada sepuluh kali pengujian sebagai pencapaian posisi optimal yang dicapai oleh anggota *quadrotor swarm* dalam melakukan

pelacakan. Data posisi pelacakan terdekat dengan titik *centroid* dinyatakan dalam Tabel 4.21.

Tabel 4. 21 Posisi Pelacakan Terdekat dengan Titik *Centroid* dengan Sepuluh Kali Pengujian

Pengujian	Pelacakan oleh Anggota *Quadrotor Swarm*	Koordinat Posisi Terdekat dengan Titik *Centroid*
Pertama	Ketiga	(0.00,-0.10)
Kedua	Pertama	(0.00,0.14)
Ketiga	Ketiga	(0.00,-0.06)
Keempat	Ketiga	(-0.01,-0.07)
Kelima	Ketiga	(-0.03, 0.00)
Keenam	Ketiga	(0.00, -0.04)
Ketujuh	Ketiga	(0.00,-0.01)
Kedelapan	Pertama	(0.00, 0.05)
Kesembilan	Pertama	(0.00, 0.05)
Kesepuluh	Pertama	(0.01, 0.03)

Dari data yang diperoleh pada Tabel 4.21 dilakukan uji varian dari sepuluh *sample* data pengujian. Perhitungan varian dari *sample* data sesuai dengan persamaan 4.1 dan hasil perhitungan varian data ditunjukkan dalam Tabel 4.12.

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1} \tag{4.1}$$

dengan,

s = varian

x_i = data ke-i

$\bar{x}$ = nilai rata-rata data

n = jumlah *sample* data

Tabel 4. 22 Nilai Varian Posisi Pelacakan Terdekat dengan Titik *Centroid* dari Sepuluh Data *Sample*

Pengujian ke-	Titik *x*	Titik *y*	Rata-rata titik *x*	Rata-rata titik *y*	$x_i - \bar{x}$	$y_i - \bar{y}$	$(x_i - \bar{x})^2$	$(y_i - \bar{y})^2$
1	0.00	-0.10	-0.003	-0.001	0.003	-0.099	0.0000	0.0098
2	0.00	0.14	-0.003	-0.001	0.003	0.141	0.0000	0.0199
3	0.00	-0.06	-0.003	-0.001	0.003	-0.059	0.0000	0.0035
4	-0.01	-0.07	-0.003	-0.001	-0.007	-0.069	0.0000	0.0048
5	-0.03	0.00	-0.003	-0.001	-0.027	0.001	0.0007	0.0000
6	0.00	-0.04	-0.003	-0.001	0.003	-0.039	0.0000	0.0015
7	0.00	-0.01	-0.003	-0.001	0.003	-0.009	0.0000	0.0001
8	0.00	0.05	-0.003	-0.001	0.003	0.051	0.0000	0.0026
9	0.00	0.05	-0.003	-0.001	0.003	0.051	0.0000	0.0026
10	0.01	0.03	-0.003	-0.001	0.013	0.031	0.0002	0.0010
						$\sum_{i=1}^{n}(x_i - \bar{x})^2$	0.0010	0.0457
						Varian = $s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$	0.0001	0.0051

Nilai varian dengan sepuluh data *sample* diketahui bahwa nilai varian untuk titik *x* adalah 0.0001 dan varian untuk titik *y* adalah 0.0051. Berdasarkan nilai varian untuk koordinat posisi yang dicapai oleh *quadrotor swarm* dalam proses pelacakan titik *centroid* menyatakan bahwa *quadrotor swarm* mampu melakukan pelacakan titik *centroid* secara optimal.

4.3 Pengujian Jarak Antar Anggota *Quadrotor Swarm* untuk Menghindari Tabrakan saat Proses Pelacakan Titik *Centroid*

Untuk menghindari tabrakan, ditentukan nilai ambang jarak antar anggota *quadrotor swarm*. Rumus untuk menentukan jarak antar anggota *quadrotor swarm* sesuai dengan Persamaan 4.2.

$$d = |x - y| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \tag{4.2}$$

dengan,

x_n : koordinat pada sumbu *x*

y_n : koordinat pada sumbu *y*

n : jumlah *quadrotor*

Jumlah anggota *quadrotor swarm* sebanyak tujuh, sehingga terdapat duapuluh satu data hubungan jarak antara *quadrotor* pertama dan *quadrotor* kedua, *quadrotor* pertama dan *quadrotor* ketiga, *quadrotor* pertama dan *quadrotor* keempat, *quadrotor* pertama dan *quadrotor* kelima, *quadrotor* pertama dan *quadrotor* keenam, *quadrotor* pertama dan *quadrotor* ketujuh, *quadrotor* kedua dan *quadrotor* ketiga, *quadrotor* kedua dan *quadrotor* keempat, *quadrotor* kedua dan *quadrotor* kelima, *quadrotor* kedua dan *quadrotor* keenam, *quadrotor* kedua dan *quadrotor* ketujuh, *quadrotor* ketiga dan *quadrotor* keempat, *quadrotor* ketiga dan *quadrotor* kelima, *quadrotor* ketiga dan *quadrotor* keenam, *quadrotor* ketiga dan *quadrotor* ketujuh, *quadrotor* keempat dan *quadrotor* kelima, *quadrotor* keempat dan *quadrotor* keenam, *quadrotor* keempat dan *quadrotor* ketujuh, *quadrotor* kelima dan *quadrotor* keenam, *quadrotor* kelima dan *quadrotor* ketujuh, *quadrotor* keenam dan *quadrotor* ketujuh. Data secara keseluruhan ditunjukkan pada Lampiran C.1 hingga Lampiran C.10.

Berdasarkan data pada Lampiran C.1 hingga Lampiran C.10, diketahui bahwa jarak masing-masing antar anggota *quadrotor swarm* tidak ada yang memiliki nilai 0 m. Sehingga dapat dinyatakan bahwa, saat melakukan proses pelacakan titik *centroid*, anggota *quadrotor swarm* mampu menghindari terjadinya tabrakan dengan anggota yang lain.

Proses menghindari terjadinya tabrakan antar anggota *quadrotor swarm* berlangsung dari iterasi pertama hingga iterasi terakhir proses pelacakan. Saat iterasi terakhir dinyatakan sebagai jarak antar anggota *quadrotor swarm*.

4.4 Jarak Antar Anggota *Quadrotor Swarm* untuk Menghindari Terjadinya Tabrakan

Berdasarkan data pengujian sebelumnya dilakukan pengambilan data untuk mengetahui kemampuan *quadrotor swarm* dalam menghindari terjadinya tabrakan. Pada pengujian ini, dilakukan pengambilan data rata-rata jarak antar anggota *quadrotor swarm* sebanyak sepuluh kali pengujian sebagai batas minimal kedekatan antar anggota *quadrotor swarm* sehingga tidak terjadi tabrakan. Data dinyatakan dalam Tabel 4.23.

Tabel 4. 23 Jarak Rata-rata Antar *Anggota Quadrotor Swarm* dengan Sepuluh Kali Pengujian

Iterasi ke-	Jarak Rata-rata Antar *Quadrotor* Pada Pengujian ke- (satuan dalam meter)									
	Satu	Dua	Tiga	Empat	Lima	Enam	Tujuh	Delapan	Sembilan	Sepuluh
1	16.06	17.60	14.54	17.05	12.42	9.49	12.04	16.07	14.77	11.76
2	15.90	17.43	14.39	16.88	12.29	9.39	11.92	15.91	14.62	11.65
3	15.74	17.25	14.25	16.71	12.17	9.30	11.80	15.75	14.47	11.53
4	15.58	17.07	14.10	16.54	12.05	9.20	11.68	15.59	14.33	11.41
5	15.42	16.90	13.95	16.37	11.92	9.11	11.56	15.43	14.18	11.29
6	15.26	16.72	13.81	16.20	11.80	9.01	11.44	15.27	14.03	11.17
7	15.10	16.55	13.66	16.03	11.67	8.92	11.32	15.11	13.88	11.06
8	14.94	16.37	13.52	15.86	11.55	8.82	11.20	14.95	13.74	10.94
9	14.78	16.19	13.37	15.69	11.42	8.73	11.08	14.79	13.59	10.82
dst	…	…	…	…	…	…	…	…	…	…
100	0.34	0.22	0.25	0.28	0.28	0.29	0.34	0.30	0.29	0.35

Dari data yang diperoleh pada Tabel 4.23 dilakukan uji varian dari sepuluh *sample* data pengujian. Perhitungan varian dari *sample* data sesuai dengan persamaan 4.1 dan hasil perhitungan varian data ditunjukkan dalam Tabel 4.24.

Tabel 4. 24 Varian Nilai Rata-rata Jarak Antar *Anggota Quadrotor Swarm* dengan Sepuluh Kali Pengujian

Pengujian ke-	Nilai Rata-rata Jarak (x)	Nilai Rata-rata ($\bar{x}$)	$(x_i - \bar{x})$	$(x_i - \bar{x})^2$
1	0.34	0.294	0.046	0.00212
2	0.22	0.294	-0.074	0.00548
3	0.25	0.294	-0.044	0.00194
4	0.28	0.294	-0.014	0.0002
5	0.28	0.294	-0.014	0.0002
6	0.29	0.294	-0.004	1.6E-05
7	0.34	0.294	0.046	0.00212
8	0.30	0.294	0.006	3.6E-05
9	0.29	0.294	-0.004	1.6E-05
10	0.35	0.294	0.056	0.00314
$\sum_{i=1}^{n}(x_i - \bar{x})^2$				0.015
Varian = $s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$				0.002

Nilai varian dengan sepuluh data *sample* diketahui bahwa nilai varian untuk nilai rata-rata jarak adalah 0.002. Berdasarkan nilai varian untuk nilai rata-rata jarak antar *quadrotor swarm* dalam proses pelacakan titik *centroid* dapat dinyatakan bahwa *quadrotor swarm* mampu menghindari terjadinya tabrakan antar anggota *quadrotor swarm*.

4.5 Pengendalian Distribusi *Quadrotor Swarm* untuk Pelacakan Titik *Centroid* Menggunakan *Modified ANNSOM* Dibandingkan dengan Metode *ANNSOM* pada [2]

Pengujian pada bagian ini bertujuan untuk mengetahui perbedaan iterasi yang diperlukan untuk pelacakan posisi titik *centroid*. Pengamatan dilakukan pada dua kondisi yaitu penggunaan metode *modified ANNSOM* dan metode *ANNSOM* pada [2]. Data pengujian ditunjukkan pada Tabel 4.25.

Tabel 4. 25 Perbedaan Iterasi untuk Pelacakan Titik *Centroid* saat Menggunakan Metode *Modified ANNSOM* dan *ANNSOM* [2]

No	Faktor Pembeda	Menggunakan Metode *Modified ANNSOM*	Menggunakan Metode *ANNSOM* [2]
1.	Iterasi yang Diperlukan	100	2000
2.	Jarak antar anggota *quadrotor swarm* (*neighbourhood*)	Varian dari sepuluh data *sample* untuk nilai rata-rata jarak adalah 0.002	Tidak diperhitungkan secara detail

4.6 Data Kecepatan Masing-masing Anggota *Quadrotor Swarm* untuk Pelacakan Titik *Centroid* Menggunakan *Modified ANNSOM*

Pengujian pada bagian ini bertujuan untuk mengetahui kecepatan masing-masing anggota *quadrotor swarm* saat melakukan pelacakan titik *centroid* dari iterasi awal hingga iterasi terakhir. Data pengujian ditunjukkan pada Tabel 4.26.

Tabel 4. 26 Data Kecepatan Masing-masing Anggota *Quadrotor Swarm* untuk Pelacakan Titik *Centroid*

Iterasi ke-	Kecepatan (m/s)						
	Q1	Q2	Q3	Q4	Q5	Q6	Q7
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.03	0.08	0.12	0.13	0.07	0.17	0.11
3	0.03	0.08	0.12	0.13	0.07	0.17	0.11
4	0.03	0.08	0.12	0.13	0.07	0.17	0.11
5	0.03	0.08	0.12	0.13	0.07	0.17	0.11
6	0.03	0.08	0.12	0.13	0.07	0.17	0.11
7	0.03	0.08	0.12	0.13	0.07	0.17	0.11
8	0.03	0.08	0.12	0.13	0.07	0.17	0.11
9	0.03	0.08	0.12	0.13	0.07	0.17	0.11
10	0.03	0.08	0.12	0.13	0.07	0.17	0.11
dst	…	…	…	…	…	…	…
93	0.03	0.08	0.12	0.13	0.07	0.17	0.11
94	0.03	0.08	0.12	0.13	0.07	0.17	0.11
95	0.03	0.08	0.12	0.13	0.07	0.17	0.11
96	0.03	0.08	0.12	0.13	0.07	0.17	0.11
97	0.03	0.11	0.12	0.13	0.07	0.17	0.11
98	0.03	0.11	0.12	0.13	0.07	0.17	0.11
99	0.03	0.08	0.12	0.20	0.19	0.17	0.11
100	0.03	0.11	0.12	0.13	0.07	0.17	0.28

Berdasarkan data pengujian Tabel 4.26, diketahui bahwa nilai kecepatan masing-masing anggota *quadrotor* adalah tetap sehingga nilai percepatannya adalah nol. Namun, pada saat anggota *quadrotor swarm* melakukan pembelokan arah untuk menghindari terjadinya tabrakan, akan terjadi perubahan nilai kecepatan.

4.7 Perhitungan Nilai *Time Constant Delay*

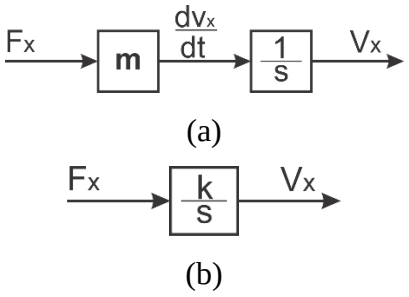
Persamaan hukum newton digunakan untuk menghitung nilai *time constant delay* yang ditunjukkan Persamaan (4.3)-(4.4).

$$F_x = m \bullet a_x = m \bullet \frac{dv_x}{dt} \tag{4.3}$$

Sehingga dapat diperoleh persamaan baru,

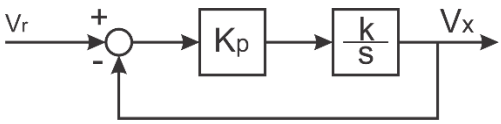
$$\frac{dv_x}{dt} = \frac{F_x}{m} \tag{4.4}$$

Persamaan (4.3)-(4.4) dapat dinyatakan dalam blok diagram yang ditunjukkan pada Gambar 4.11.



Gambar 4. 11 (a) Blok Diagram yang Menunjukkan Persamaan Hukum Newton; (b) Blok Diagram yang Menunjukkan Persamaan Hukum Newton dengan *Massa* Dinyatakan Sebagai Konstanta

Berdasarkan blok diagram pada Gambar 4.11 dan Persamaan (4.4) dapat diperoleh blok diagram yang ditunjukkan pada Gambar 4.12 dan Persamaan (4.5)-(4.6) untuk menghitung nilai *time constant delay*.



Gambar 4. 12 Blok Diagram untuk Menghitung Nilai *Time Constant Delay*

$$v_x(s) = \frac{K_p \bullet \frac{k}{s}}{1 + K_p \bullet \frac{k}{s}} v_r = \frac{1}{\frac{1}{K_p \bullet k} s + 1} v_r = \frac{1}{\tau s + 1} v_r \tag{4.5}$$

Sehingga dapat dinyatakan bahwa,

$$\tau = \frac{1}{K_p \bullet k} \tag{4.6}$$

Jika nilai $K_p = 0.2$ dan $Massa = m = k = 10kg$, maka dapat diperoleh nilai *time constant delay* $\tau = \frac{1}{K_p \bullet k} = \frac{1}{0.2 \bullet 10} = \frac{1}{2} = 0.5$ detik.

BAB 5
PENUTUP

5.1 Kesimpulan

Berdasarkan pengujian yang telah dilakukan dapat disimpulkan bahwa,

1. Berdasarkan nilai varian koordinat posisi yang dicapai oleh *quadrotor swarm* dalam proses pelacakan titik *centroid* dengan sepuluh data *sample* diketahui bahwa nilai varian untuk titik *x* adalah 0.0001 dan varian untuk titik *y* adalah 0.0051 dengan *time constant delay* 0.5 detik. Dari data ini dapat disimpulkan bahwa *quadrotor swarm* mampu melakukan pelacakan titik *centroid* secara optimal.
2. Berdasarkan varian nilai rata-rata jarak antar *quadrotor swarm* dalam proses pelacakan titik *centroid* dengan sepuluh data *sample* diketahui bahwa nilai rata-rata jarak adalah 0.002. Sehingga dapat dinyatakan bahwa *quadrotor swarm* mampu menghindari terjadinya tabrakan antar anggota *quadrotor swarm*.
3. Penggunaan metode *modified ANNSOM* untuk pengendalian distribusi *quadrotor swarm* dalam proses pelacakan titik *centroid* memerlukan iterasi yang lebih sedikit (lebih cepat) jika dibandingkan dengan metode *ANNSOM*.

5.2 Saran

Penelitian ini dapat dikembangkan dengan menambahkan algoritma untuk membentuk formasi yang berbeda-beda dalam dimensi ruang, sehingga dapat diperoleh pola yang paling efektif untuk menampung jumlah *quadrotor* yang banyak.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] IMingzheng, Liu, Li Hongjian, and Zhai Hualei. "Unmanned Aerial Vehicles for Logistics Applications." Control Conference (CCC), 2014 33rd Chinese. IEEE, 2014.

[2] Maningo, Jose Martin Z., et al. "Obstacle Avoidance for Quadrotor Swarm Using Artificial Neural Network Self-Organizing Map." Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), 2015 International Conference on. IEEE, 2015.

[3] Nakano, Reiichiro Christian S., et al. "A Genetic Algorithm Approach to Swarm Centroid Tracking in Quadrotor Unmanned Aerial Vehicles." Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), 2014 International Conference on. IEEE, 2014.

[4] Bandala, Argel A., et al. "Implementation of Varied Particle Container for Smoothed Particle Hydrodynamics-Based Aggregation for Unmanned Aerial Vehicle Quadrotor Swarm." Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on. IEEE, 2016.

[5] Santoso, Budi dan Willy, P. ,Metode Metaheuristik Konsep dan Implementasi, Guna Widya, Surabaya.2001.

[6] Hornik, Maxwell Stinchcombe, Halbert White, Multilayer feedforward networks are universal approximators, Neural Networks, Volume 2, Issue 5, 1989, Pages 359-366, ISSN 0893-6080

[7] Bandala, A., R. Vicerra, and E. Dadios. "Swarming Algorithm for Unmanned Aerial Vehicle (UAV) Quadrotors in Swarm Behavior for Aggregation Foraging Formation and Tracking." Journal of Advanced Computational Intelligence and Intelligent Informatics 8.9, 2014.

[8] Gazi, Veysel, and Kevin M. Passino. Swarm stability and optimization. Springer Science & Business Media, 2011.

[9] Prasetyo, A. P. (2015). Implementasi Navigasi *Multiple Autonomous Mobile Robot* untuk Mencari Sumber Gas dengan Menggunakan Metode FKN-PSO (*Fuzzy Kohonen Network - Particle Swarm Optimization*), (Doctoral dissertation, Institut Teknologi Sepuluh Nopember).

[10] Setyawan, N. (2017). Pengembangan Adaptive Particle Swarm Optimization (PSO) Dan Aplikasinya Pada Perencanaan Jalur Mobile Robot Dengan Halangan Dinamis (Doctoral dissertation, Institut Teknologi Sepuluh Nopember).

LAMPIRAN

Program Utama

```
% TEST  --  Modified ANNSOM  -- Artificial Neural Network
Termodifikasi--

clc; clear; clear global; clf; %figure(200);

% Set up problem for quadratic Bowl
% Minimize this function BENTUK MAP
objFun = @(x)(sum(x.^2,1));


xLow = -15*ones(2,1); % lower bound on the search space = batas
area maksimal yg dihuni swarm pada sumbu x
xUpp = 15*ones(2,1); % upper bound on the search space = batas
area maksimal yg dihuni swarm pada sumbu y
x0 = [ ];  % inisialisai awal x0

options.alpha = 0.4;  % weight on current search direction
options.beta = 0.9;   % weight on local best search direction
options.gamma = 0.9;  % weight on global best search direction

options.nPopulation = 5;% jumlah populasi/ anggota swarm
options.maxIter = 100; % jumlah iterasi maksimal yang dibatasi,
iterasi berhenti jika anggota swarm telah berada di posisi
centroid

%BAGIAN PLOTTING GAMBAR
options.plotFun = @plotBowl;

%%%% Solve diambil dari ANNSOM.m
[xBest, fBest, info, dataLog] = ANNSOM(objFun, x0, xLow, xUpp,
options);
%ANNSOM2-> pakai persamaan quadrotor
%ANNSOM-> pakai persamaan partikel

% xBest= koordinat terakhir anggota swarm; fBest=bobot; info=untuk
% menggambar; dataLog=diambil dari info=== hasil ini diperoleh
dari
% perhitungan Program PSO

%%%% Analysis
%figure(101);
figure(201); clf;% untuk menampilkan gambar
plotPsoHistory(info);%untuk nge plot gambar berdasarkan data yang
tersimpan di info
```

A. 2. Program untuk Menggambar objek (quadrotor dll)

```
function plotBowl(dataLog, iter)

%digunakan untuk menggambar posisi dari partikel
figure(100);
hold on;
axis([-5,5,-5,5]); axis equal;
%%%% gambar lingkaran…mulai dari terendah hingga tertinggi….
rectangle('Position',[-5,-
5,10,10],'Curvature',[1,1],'LineWidth',1);
rectangle('Position',0.8^2*[-5,-
5,10,10],'Curvature',[1,1],'LineWidth',1);
rectangle('Position',0.6^2*[-5,-
5,10,10],'Curvature',[1,1],'LineWidth',1);
rectangle('Position',0.4^2*[-5,-
5,10,10],'Curvature',[1,1],'LineWidth',1);
rectangle('Position',0.2^2*[-5,-
5,10,10],'Curvature',[1,1],'LineWidth',1);

rectangle('Position',[-1,-1,2,2],'Curvature',[1,1],'LineWidth',1);
rectangle('Position',0.8^2*[-1,-
1,2,2],'Curvature',[1,1],'LineWidth',1);
rectangle('Position',0.6^2*[-1,-
1,2,2],'Curvature',[1,1],'LineWidth',1);
rectangle('Position',0.4^2*[-1,-
1,2,2],'Curvature',[1,1],'LineWidth',1);
rectangle('Position',0.2^2*[-1,-
1,2,2],'Curvature',[1,1],'LineWidth',1);

%posisi koordinat partikel
x = dataLog.X_Best(1,:);   %First dimension
y = dataLog.X_Best(2,:);   %Second dimension

plot(x(1),y(1),'ro','MarkerSize',5,'LineWidth',0.5);  %quadcopter
pertama merah
plot(x(2),y(2),'bo','MarkerSize',5,'LineWidth',0.5);  %quadcopter
kedua biru
plot(x(3),y(3),'yo','MarkerSize',5,'LineWidth',0.5);  %quadcopter
ketiga kuning
plot(x(4),y(4),'go','MarkerSize',5,'LineWidth',0.5);  %quadcopter
keempat hijau
plot(x(5),y(5),'ko','MarkerSize',5,'LineWidth',0.5);  %quadcopter
kelima hitam

title(sprintf('Iteration: %d, iter));
xlabel('x1')
ylabel('x2');

%%%% Force drawing:
drawnow;
pause(1);  %Make it possible to see updates


end
```

A. 3. Sub Program untuk Modified ANN SOM

```
function [xBest, fBest, info, dataLog] = ANNSOM(objFun, x0, xLow,
xUpp, options)

%%%% Basic input validation:

[n, m] = size(xLow);
if m ~= 1
    error('x0 is not a valid size! Must be a column vector.')
end
[nRow, nCol] = size(xLow);
if nRow ~= n || nCol ~= 1
    error(['xLow is not a valid size! Must be [' num2str(n) ',
1]']);
end
[nRow, nCol] = size(xUpp);
if nRow ~= n || nCol ~= 1
    error(['xUpp is not a valid size! Must be [' num2str(n) ',
1]']);
end

%%%% Options Struct:
default.alpha = 0.6;                % search weight on current
search direction
default.beta = 0.9;                 % search weight on global best
default.gamma = 0.9;                % search weight on local best
default.nPopulation = 3*n;          % 3*n = population count
default.maxIter = 100;              % maximum number of
generations
default.tolFun = 1e-6;              % exit when variance in
objective is < tolFun
default.tolX = 1e-10;               % exit when norm of variance
in state < tolX
default.flagVectorize = false;      % is the objective function
vectorized?
default.flagMinimize = true;        % true for minimization, false
for maximization
default.xDelMax = xUpp - xLow;      % Maximnum position update;
default.flagWarmStart = false;      % Directly use the initial
point?
default.guessWeight = 0.2;          % on range [0, 0.9);  0 =
ignore guess,  1 = start at guess
default.plotFun = [];               % Handle to a function for
plotting the progress
default.display = 'iter';           % Print out progress to user
default.printMod = 1;               % Print out every [printMod]
iterations

if nargin == 5  % user provided options struct!
    options = mergeOptions(default,options);
else% no user-defined options. Use defaults.
    options = default;
end
```

```
%%% Options validation:
if options.guessWeight < 0
    options.guessWeight = 0;
    disp('WARNING: options.guessWeight must be on range [0,
0.9)');
elseif options.guessWeight > 0.9
    options.guessWeight = 0.9;
    disp('WARNING: options.guessWeight must be on range [0,
0.9)');
end

%%%% Minimize vs Maximize:
if options.flagMinimize
    optFun = @min;
else
    optFun = @max;
end

%%%% Check to see if user defined x0. If not, force defaults
if isempty(x0)
   x0 = 0.5*xLow + 0.5*xUpp;
   options.guessWeight = 0.0;
   options.flagWarmStart = false;
end

%%%% Initialize the population

% Sample two random points in the search space for each particle
m = options.nPopulation;  %population size
X1 = xLow*ones(1,m) + ((xUpp-xLow)*ones(1,m)).*rand(n,m);
X2 = xLow*ones(1,m) + ((xUpp-xLow)*ones(1,m)).*rand(n,m);

% Move initial points towards initial guess, by convex combination
w = options.guessWeight;  %for initialization
X0 = x0*ones(1,m);
X1 = w*X0 + (1-w)*X1;
X2 = w*X0 + (1-w)*X2;

% Sample two random points in the search space for each particle
m = options.nPopulation;  %population size
XX1 = xLow*ones(1,m) + ((xUpp-xLow)*ones(1,m)).*rand(n,m);
XX2 = xLow*ones(1,m) + ((xUpp-xLow)*ones(1,m)).*rand(n,m);

% Move initial points towards initial guess, by convex combination
w = options.guessWeight;  %for initialization
XX0 = x0*ones(1,m);
XX1 = w*X0 + (1-w)*XX1;
XX2 = w*X0 + (1-w)*XX2;
%================================tambah NN
================================

inputs = [0.9 0.9 0.2 0.1 -0.2 -0.3 -0.9;
          0.9 0.9 0.2 0.1 -0.2 -0.3 -0.9;
          0.9 0.9 0.2 0.1 -0.2 -0.3 -0.9;
          0.9 0.9 0.2 0.1 -0.2 -0.3 -0.9;
          0.9 0.9 0.2 0.1 -0.2 -0.3 -0.9;];
```

```
targets = [0.4 0.2 0.1 0 -0.1 -0.2 -0.4];

% Create a Fitting Network
hiddenLayerSize = 10;
net = fitnet(hiddenLayerSize);

% Set up Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 35/100;
net.divideParam.testRatio = 35/100;

% Train the Network
[net,tr] = train(net,inputs,targets);

%tes1 = [0.1 0.2;0.3 0.4];

input1=[X1(1,1) X1(1,2) X1(1,3) X1(1,4) X1(1,5)]';
input2=[X1(1,1) X1(1,2) X1(1,3) X1(1,4) X1(1,5)]';

%input2=[X1(1,1) X1(1,2)]';
% Test the Network
%input2 = coba;

outputs = net(input1);%hasil yg sudah ditraning x
X1(1,1)=outputs;
outputs = net(input2);%y
X1(1,2)=outputs;

%=========================================================================
========
% Initialize population:
X = X1;      % Initial position of the population
V = X2-X1;  % Initial "velocity" of the population

% inisialisasi untuk quadrotor ke 2
XX = XX1;      % Initial position of the population
V2 = XX2-XX1;  % Initial "velocity" of the population

% Check for warm start. If so, override random initial point with
x0
if options.flagWarmStart
   X(:,1) = x0;
   V(:,1) = zeros(size(x0));
end
% Check for warm start. If so, override random initial point with
x0
if options.flagWarmStart
   XX(:,1) = x0;
   V2(:,1) = zeros(size(x0));
end
if options.flagVectorize   % Batch process objective
    X_Low = xLow*ones(1,m);
    X_Upp = xUpp*ones(1,m);
    F = objFun(X);  % Function value at each particle in the
population
else% Objective not vectorized
```

```
    F = zeros(1,m);
for idx = 1:m   % Loop over particles
        F(1,idx) = objFun(X(:,idx));
end

% nilai fitness untuk quadrotor kedua...
    F2 = zeros(1,m);
for idx = 1:m   % Loop over particles
        F2(1,idx) = objFun(XX(:,idx));
end
end

X_Best = X;  % Best point, for each particle in the population
F_Best = F;  % Value of best point, for each particle in the
population

%nilai x best dan fbest untuk quadrotor kedua

X_Best2 = XX;  % Best point, for each particle in the population
F_Best2 = F2;  % Value of best point, for each particle in the
population


[F_Global, I_Global] = optFun(F_Best); % Value of best point ever,
over all points
X_Global = X(:, I_Global);%[2;-2]; % Best point ever, over all
points

[F_Global2, I_Global2] = optFun(F_Best2); % Value of best point
ever, over all points
X_Global2 = XX(:, I_Global2);%[2;-2]; % Best point ever, over all
points


%%%% Allocate memory for the dataLog
maxIter = options.maxIter;
dataLog(maxIter) = makeStruct(X,XX, V,V2, F,F2, X_Best, X_Best2,
F_Best, F_Best2, X_Global, X_Global2, F_Global,F_Global2,
I_Global,I_Global2);

batas1 = zeros(maxIter);
batas2 = zeros(maxIter);
batas3 = zeros(maxIter);
batas4 = zeros(maxIter);
batas5 = zeros(maxIter);
batas6 = zeros(maxIter);
batas7 = zeros(maxIter);
batas8 = zeros(maxIter);
batas9 = zeros(maxIter);
batas10 = zeros(maxIter);
batastot = zeros(maxIter);

SimpanX_Global_X = zeros(maxIter);
SimpanX_Global_Y = zeros(maxIter);
SimpanX_Global2_X = zeros(maxIter);
```

```
SimpanX_Global2_Y = zeros(maxIter);

info.X_Global = zeros(n,maxIter);
info.X_Global2 = zeros(n,maxIter);
info.F_Global = zeros(1,maxIter);
info.F_Global2 = zeros(1,maxIter);
info.I_Global = zeros(1,maxIter);
info.I_Global2 = zeros(1,maxIter);
info.X_Best_Var = zeros(n,maxIter);
info.F_Best_Var = zeros(1,maxIter);
info.X_Best_Mean = zeros(n,maxIter);
info.F_Best_Mean = zeros(1,maxIter);
info.X_Var = zeros(n,maxIter);
info.X_Var2 = zeros(n,maxIter);
info.F_Var = zeros(1,maxIter);
info.X_Mean = zeros(n,maxIter);
info.F_Mean = zeros(1,maxIter);
info.iter = 1:maxIter;

%%%% MAIN LOOP:
info.exitFlag = 1;   %Assume that we will reach maximum iteration
for iter = 1:maxIter

%%% Compute new generation of points:
if iter > 1   % Then do an update on each particle

        r1 = rand(n,m);
        r2 = rand(n,m);
        r3 = rand(n,m);
        r4 = rand(n,m);

if options.flagVectorize   % Batch process objective

            V =  ...   %Update equations
                options.alpha*V + ...    % Current search
direction
                options.beta*r1.*((X_Global*ones(1,m))-X) + ...  %
Global direction
                options.gamma*r2.*(X_Best-X);    % Local best
direction
            X_New = X + V;  % Update position
            X = max(min(X_New, X_Upp), X_Low);   % Clamp position
to bounds

            F = objFun(X);   %Evaluate

            F_Best_New = optFun(F_Best, F);   %Compute the best
point
            idxUpdate = F_Best_New ~= F_Best;  % Which indicies
updated?
            X_Best(:,idxUpdate) = X(:,idxUpdate);  %Copy over new
best points
            F_Best = F_Best_New;
            [F_Global, I_Global] = optFun(F_Best); % Value of best
point ever, over all points
```

```
            X_Global = X(:, I_Global); % Best point ever, over all
points

%==================================
              V2 =  ...   %Update equations
                options.alpha*V2 + ...    % Current search
direction
                options.beta*r3.*((X_Global2*ones(1,m))-XX) + ...
% Global direction
                options.gamma*r4.*(X_Best2-XX);    % Local best
direction
            X_New2 = XX + V2;  % Update position
            XX = max(min(X_New2, X_Upp), X_Low);   % Clamp
position to bounds

            F2 = objFun(XX);   %Evaluate

            F_Best_New2 = optFun(F_Best2, F2);   %Compute the best
point
            idxUpdate2 = F_Best_New2 ~= F_Best2;  % Which indicies
updated?
            X_Best2(:,idxUpdate2) = XX(:,idxUpdate2);  %Copy over
new best points
            F_Best2 = F_Best_New2;
            [F_Global2, I_Global2] = optFun(F_Best2); % Value of
best point ever, over all points
            X_Global2 = XX(:, I_Global2); % Best point ever, over
all  points
%==================================

else%Objective is not vectorized.

for idx = 1:m   %%%%%%% Loop over particles  %%%%%%%%%%%%%%

                V(:,idx) =  ...   %Update equations
                    options.alpha*V(:,idx) + ...    % Current
search direction
                    options.beta*r1(:,idx).*(X_Global-X(:,idx)) +
...  % Global direction
                    options.gamma*r2(:,idx).*(X_Best(:,idx)-
X(:,idx));    % Local best direction

                X_New = X(:,idx) + V(:,idx);  % Update position
                X(:,idx) = max(min(X_New, xUpp), xLow);   % Clamp
position to bounds

                F(:,idx) = objFun(X(:,idx));   %Evaluate

                 V2(:,idx) =  ...   %Update equations
                    options.alpha*V2(:,idx) + ...    % Current
search direction
                    options.beta*r3(:,idx).*(X_Global2-XX(:,idx))
+ ...  % Global direction
                    options.gamma*r4(:,idx).*(X_Best2(:,idx)-
XX(:,idx));    % Local best direction
```

```
                X_New2 = XX(:,idx) + V2(:,idx);  % Update position
                XX(:,idx) = max(min(X_New2, xUpp), xLow);    %
Clamp position to bounds

                F2(:,idx) = objFun(XX(:,idx));   %Evaluate

                [F_Best(1,idx), iBest] = optFun([F(1,idx),
F_Best(1,idx)]);

                [F_Best2(1,idx), iBest2] = optFun([F2(1,idx),
F_Best2(1,idx)]);

if iBest == 1  %Then new point is better!
                    X_Best(:,idx) = X(:,idx);
                    [F_Global, iBest] = optFun([F_Best(1,idx),
F_Global]);
if iBest == 1 %Then new point is the global best!
                        X_Global = X_Best(:,idx);
end
end

if iBest2 == 1  %Then new point is better!
                    X_Best2(:,idx) = XX(:,idx);
                    [F_Global2, iBest2] = optFun([F_Best2(1,idx),
F_Global2]);
if iBest2 == 1 %Then new point is the global best!
                        X_Global2 = X_Best2(:,idx);
end
end

end%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end
end

%digunakan sebagai penyimpan data koordinat..untuk
menggambar

    SimpanX_Global_1(iter,1) = X_Best(1,1);
    SimpanX_Global_1(iter,2) = X_Best(2,1);

    SimpanX_Global_2(iter,1) = X_Best(1,2);
    SimpanX_Global_2(iter,2) = X_Best(2,2);

    SimpanX_Global_3(iter,1) = X_Best(1,3);
    SimpanX_Global_3(iter,2) = X_Best(2,3);

    SimpanX_Global_4(iter,1) = X_Best(1,4);
    SimpanX_Global_4(iter,2) = X_Best(2,4);

    SimpanX_Global_5(iter,1) = X_Best(1,5);
    SimpanX_Global_5(iter,2) = X_Best(2,5);
%cla reset;
if iter>1
```

```
%digunakan untuk menggambar garis dari suatu iterasi ke
iterasi yang lain

        line([SimpanX_Global_1(iter-1,1)
SimpanX_Global_1(iter,1)],[SimpanX_Global_1(iter-1,2)
SimpanX_Global_1(iter,2)]);
        line([SimpanX_Global_2(iter-1,1)
SimpanX_Global_2(iter,1)],[SimpanX_Global_2(iter-1,2)
SimpanX_Global_2(iter,2)]);
        line([SimpanX_Global_3(iter-1,1)
SimpanX_Global_3(iter,1)],[SimpanX_Global_3(iter-1,2)
SimpanX_Global_3(iter,2)]);
        line([SimpanX_Global_4(iter-1,1)
SimpanX_Global_4(iter,1)],[SimpanX_Global_4(iter-1,2)
SimpanX_Global_4(iter,2)]);
        line([SimpanX_Global_5(iter-1,1)
SimpanX_Global_5(iter,1)],[SimpanX_Global_5(iter-1,2)
SimpanX_Global_5(iter,2)]);
end

% drawnow;
% figure(100);

%%% Log Data DATALOG
    dataLog(iter) = makeStruct(X,XX, V,V2, F,F2, X_Best,X_Best2,
F_Best,F_Best2, X_Global,X_Global2, F_Global,F_Global2,
I_Global,I_Global2);
    info.X_Global(:,iter) = X_Global; %posisi Q1 tiap iterasi
    info.X_Global2(:,iter) = X_Global2; %posisi Q2 tiap iterasi
    info.F_Global(iter) = F_Global;
    info.F_Global2(iter) = F_Global2;
    info.I_Global(iter) = I_Global;
    info.I_Global2(iter) = I_Global2;
    info.X_Var(:,iter) = var(X, 0, 2);
    info.X_Best_Var(:,iter) = var(X_Best, 0, 2);
    info.X_Mean(:,iter) = mean(X, 2);
    info.X_Best_Mean(:,iter) = mean(X_Best, 2);
    info.F_Var(1,iter) = var(F);
    info.F_Best_Var(1,iter) = var(F_Best);
    info.F_Mean(1,iter) = mean(F);
    info.F_Best_Mean(1,iter) = mean(F_Best);

%%% Plot
if ~isempty(options.plotFun)
        options.plotFun(dataLog(iter), iter);
end

%%% Print:
    xVar = norm(info.X_Var(:,iter));
if strcmp('iter',options.display)
if mod(iter-1,options.printMod)==0
           fprintf('iter: %3d,   fBest: %9.3e,   fVar: %9.3e
xVar: %9.3e  \n',...
               iter, info.F_Global(iter), info.F_Var(1,iter),
xVar);
end
```

```
end

%xVar2 = norm(info.X_Var(:,iter));
%%% Convergence:

%=====jarak antar quadrotor minimal===
%batas1(iter) =sqrt(((X_Global(2)-X_Global(2))*(X_Global(2)-
X_Global2(2)))+((X_Global(1)-X_Global2(1))*(X_Global(1)-
X_Global2(1))));
%menghitung batas minimal pada quadrotor

    batas1(iter) =sqrt(((X_Best(1,1)-X_Best(1,2))*(X_Best(1,1)-
X_Best(1,2)))+((X_Best(2,1)-X_Best(2,2))*(X_Best(2,1)-
X_Best(2,2))));
    batas2(iter) =sqrt(((X_Best(1,1)-X_Best(1,3))*(X_Best(1,1)-
X_Best(1,3)))+((X_Best(2,1)-X_Best(2,3))*(X_Best(2,1)-
X_Best(2,3))));
    batas3(iter) =sqrt(((X_Best(1,1)-X_Best(1,4))*(X_Best(1,1)-
X_Best(1,4)))+((X_Best(2,1)-X_Best(2,4))*(X_Best(2,1)-
X_Best(2,4))));
    batas4(iter) =sqrt(((X_Best(1,1)-X_Best(1,5))*(X_Best(1,1)-
X_Best(1,5)))+((X_Best(2,1)-X_Best(2,5))*(X_Best(2,1)-
X_Best(2,5))));

    batas5(iter) =sqrt(((X_Best(1,2)-X_Best(1,3))*(X_Best(1,2)-
X_Best(1,3)))+((X_Best(2,2)-X_Best(2,3))*(X_Best(2,2)-
X_Best(2,3))));
    batas6(iter) =sqrt(((X_Best(1,2)-X_Best(1,4))*(X_Best(1,2)-
X_Best(1,4)))+((X_Best(2,2)-X_Best(2,4))*(X_Best(2,2)-
X_Best(2,4))));
    batas7(iter) =sqrt(((X_Best(1,2)-X_Best(1,5))*(X_Best(1,2)-
X_Best(1,5)))+((X_Best(2,2)-X_Best(2,5))*(X_Best(2,2)-
X_Best(2,5))));

    batas8(iter) =sqrt(((X_Best(1,3)-X_Best(1,4))*(X_Best(1,3)-
X_Best(1,4)))+((X_Best(2,3)-X_Best(2,4))*(X_Best(2,3)-
X_Best(2,4))));
    batas9(iter) =sqrt(((X_Best(1,3)-X_Best(1,5))*(X_Best(1,3)-
X_Best(1,5)))+((X_Best(2,3)-X_Best(2,5))*(X_Best(2,3)-
X_Best(2,5))));

    batas10(iter)=sqrt(((X_Best(1,4)-X_Best(1,5))*(X_Best(1,4)-
X_Best(1,5)))+((X_Best(2,4)-X_Best(2,5))*(X_Best(2,4)-
X_Best(2,5))));


    batastot(iter)=
(batas1(iter)+batas2(iter)+batas3(iter)+batas4(iter)+batas5(iter)+
batas6(iter)+batas7(iter)+batas8(iter)+batas9(iter)+batas10(iter))
/10;


if info.F_Var(1,iter) < options.tolFun
        info.exitFlag = 0;
        dataLog = dataLog(1:iter);
```

```
        info = truncateInfo(info,maxIter,iter);
break

elseif xVar < options.tolX
        info.exitFlag = 2;
        dataLog = dataLog(1:iter);
        info = truncateInfo(info,maxIter,iter);
break
%===batasan jk quadrotor jaraknya 0.5====
elseif batastot(iter) < 1
            info.exitFlag = 2;
            dataLog = dataLog(1:iter);
            info = truncateInfo(info,maxIter,iter);
break
%end
end
end

xBest = info.X_Global(:,end);
fBest = info.F_Global(end);
info.input = makeStruct(objFun, x0, xLow, xUpp, options);  %Copy
inputs
info.fEvalCount = iter*m;

%%% Print:
if strcmp('iter',options.display) ||
strcmp('final',options.display)
switch info.exitFlag
case 0
            fprintf('Optimization Converged. Exit: fVar < tolFun
\n');
case 1
            fprintf('Maximum iteration reached. \n');
case 2
            fprintf('Optimization Converged. Exit: norm(xVar) <
tolX \n');
end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%

function info = truncateInfo(info,maxIter,iter)
%
% Removes the empty entries in the info struct

names = fieldnames(info);
for i=1:length(names)
if (isnumeric(info.(names{i})))    % Check if it's a matrix
if size(info.(names{i}),2) == maxIter    % Check if it is
iteration data
            info.(names{i}) = info.(names{i})(:,1:iter);
```

```
end
end
end
end
```

A. 4. Sub Program *Modified ANN SOM* dengan Persamaan *Quadrotor*

```
function [xBest, fBest, info, dataLog] = ANNSOM2(objFun, x0, xLow,
xUpp, options)
%

%%%% Basic input validation:

[n, m] = size(xLow);
if m ~= 1
    error('x0 is not a valid size! Must be a column vector.')
end
[nRow, nCol] = size(xLow);
if nRow ~= n || nCol ~= 1
    error(['xLow is not a valid size! Must be [' num2str(n) ',
1]']);
end
[nRow, nCol] = size(xUpp);
if nRow ~= n || nCol ~= 1
    error(['xUpp is not a valid size! Must be [' num2str(n) ',
1]']);
end

%%%% Options Struct:
default.alpha = 0.6;                % search weight on current
search direction
default.beta = 0.9;                 % search weight on global best
default.gamma = 0.9;                % search weight on local best
default.nPopulation = 3*n;          % 3*n = population count
default.maxIter = 100;              % maximum number of
generations
default.tolFun = 1e-6;              % exit when variance in
objective is < tolFun
default.tolX = 1e-10;               % exit when norm of variance
in state < tolX
default.flagVectorize = false;      % is the objective function
vectorized?
default.flagMinimize = true;        % true for minimization, false
for maximization
default.xDelMax = xUpp - xLow;      % Maximnum position update;
default.flagWarmStart = false;      % Directly use the initial
point?
default.guessWeight = 0.2;          % on range [0, 0.9);  0 =
ignore guess,  1 = start at guess
default.plotFun = [];               % Handle to a function for
plotting the progress
default.display = 'iter';           % Print out progress to user
default.printMod = 1;               % Print out every [printMod]
iterations

if nargin == 5  % user provided options struct!
```

```
    options = mergeOptions(default,options);
else% no user-defined options. Use defaults.
    options = default;
end

%%% Options validation:
if options.guessWeight < 0
    options.guessWeight = 0;
    disp('WARNING: options.guessWeight must be on range [0,
0.9)');
elseif options.guessWeight > 0.9
    options.guessWeight = 0.9;
    disp('WARNING: options.guessWeight must be on range [0,
0.9)');
end

%%%% Minimize vs Maximize:
if options.flagMinimize
    optFun = @min;
else
    optFun = @max;
end

%%%% Check to see if user defined x0. If not, force defaults
if isempty(x0)
   x0 = 0.5*xLow + 0.5*xUpp;
   options.guessWeight = 0.0;
   options.flagWarmStart = false;
end

%%%% Initialize the population

% Sample two random points in the search space for each particle
m = options.nPopulation;  %population size
X1 = xLow*ones(1,m) + ((xUpp-xLow)*ones(1,m)).*rand(n,m);
X2 = xLow*ones(1,m) + ((xUpp-xLow)*ones(1,m)).*rand(n,m);

% Move initial points towards initial guess, by convex combination
w = options.guessWeight;  %for initialization
X0 = x0*ones(1,m);
X1 = w*X0 + (1-w)*X1;
X2 = w*X0 + (1-w)*X2;

% Sample two random points in the search space for each particle
m = options.nPopulation;  %population size
XX1 = xLow*ones(1,m) + ((xUpp-xLow)*ones(1,m)).*rand(n,m);
XX2 = xLow*ones(1,m) + ((xUpp-xLow)*ones(1,m)).*rand(n,m);

% Move initial points towards initial guess, by convex combination
w = options.guessWeight;  %for initialization
XX0 = x0*ones(1,m);
XX1 = w*X0 + (1-w)*XX1;
XX2 = w*X0 + (1-w)*XX2;
%================================tambah NN
================================
```

```
inputs = [0.9 0.9 0.2 0.1 -0.2 -0.3 -0.9;
          0.9 0.9 0.2 0.1 -0.2 -0.3 -0.9;
          0.9 0.9 0.2 0.1 -0.2 -0.3 -0.9;
          0.9 0.9 0.2 0.1 -0.2 -0.3 -0.9;
          0.9 0.9 0.2 0.1 -0.2 -0.3 -0.9;];
targets = [0.4 0.2 0.1 0 -0.1 -0.2 -0.4];

% Create a Fitting Network
% JUMLAH LAYER SIZE
hiddenLayerSize = 10;
net = fitnet(hiddenLayerSize);

% Set up Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 35/100;
net.divideParam.testRatio = 35/100;

% Train the Network
[net,tr] = train(net,inputs,targets);

%tes1 = [0.1 0.2;0.3 0.4];

input1=[X1(1,1) X1(1,2) X1(1,3) X1(1,4) X1(1,5)]';
input2=[X1(1,1) X1(1,2) X1(1,3) X1(1,4) X1(1,5)]';

outputs = net(input1);%hasil yg sudah ditraning x
X1(1,1)=outputs;
outputs = net(input2);%y
X1(1,2)=outputs;
%errors = gsubtract(outputs,targets);
%performance = perform(net,targets,outputs)

% View the Network
%view(net)

%=================================================================
========
% Initialize population:
X = X1;     % Initial position of the population
V = X2-X1;  % Initial "velocity" of the population

% inisialisasi untuk quadrotor ke 2
XX = XX1;     % Initial position of the population
V2 = XX2-XX1;  % Initial "velocity" of the population

% Check for warm start. If so, override random initial point with
x0
if options.flagWarmStart
   X(:,1) = x0;
   V(:,1) = zeros(size(x0));
end
% Check for warm start. If so, override random initial point with
x0
if options.flagWarmStart
   XX(:,1) = x0;
   V2(:,1) = zeros(size(x0));
```

```
end
if options.flagVectorize   % Batch process objective
    X_Low = xLow*ones(1,m);
    X_Upp = xUpp*ones(1,m);
    F = objFun(X);  % Function value at each particle in the
population
else% Objective not vectorized
    F = zeros(1,m);
for idx = 1:m   % Loop over particles
        F(1,idx) = objFun(X(:,idx));
end

% nilai fitness untuk quadrotor kedua...
    F2 = zeros(1,m);
for idx = 1:m   % Loop over particles
        F2(1,idx) = objFun(XX(:,idx));
end
end

X_Best = X;  % Best point, for each particle in the population
F_Best = F;  % Value of best point, for each particle in the
population


  roll=10;
  pitch=10;
  yaw=10;

  u1=80;
  m=10;

%Transfer Func Quadrotor Sumbu X

pers1=(sin(roll/180*3.14)*sin(pitch/180*3.14)+cos(yaw/180*3.14)*si
n(roll/180*3.14)*cos(pitch/180*3.14))*u1/(m);

%Transfer Func Quadrotor Sumbu y
 pers2=(-
sin(roll/180*3.14)*cos(pitch/180*3.14)+cos(yaw/180*3.14)*sin(roll/
180*3.14)*sin(pitch/180*3.14))*u1/m;


//digunakana untuk mengetahui posisi awal dari
koordinat x dan y setiap quadrotot
selanjutnya…x_best(n,m)…n->1 =➔ sumbu x, n->2 sumbu y

 X_Best(1,1)=X_Best(1,1)+pers1;
 X_Best(2,1)=X_Best(2,1)+pers2;

 X_Best(1,1)=X_Best(1,2)+pers1;
 X_Best(2,1)=X_Best(2,2)+pers2;

 X_Best(1,3)=X_Best(1,3)+pers1;
 X_Best(2,3)=X_Best(2,3)+pers2;
```

```
 X_Best(1,4)=X_Best(1,4)+pers1;
 X_Best(2,4)=X_Best(2,4)+pers2;

 X_Best(1,5)=X_Best(1,5)+pers1;
 X_Best(2,5)=X_Best(2,5)+pers2;

 simpandata1=X_Best(1,1);
 simpandata2=X_Best(2,1);

 simpandata3=X_Best(1,2);
 simpandata4=X_Best(2,2);

 simpandata5=X_Best(1,3);
 simpandata6=X_Best(2,3);

 simpandata7=X_Best(1,4);
 simpandata8=X_Best(2,4);

 simpandata9=X_Best(1,5);
 simpandata10=X_Best(2,5);

%nilai x best dan fbest untuk quadrotor kedua

X_Best2 = XX;  % Best point, for each particle in the population
F_Best2 = F2;  % Value of best point, for each particle in the
population


[F_Global, I_Global] = optFun(F_Best); % Value of best point ever,
over all points
X_Global = X(:, I_Global);%[2;-2]; % Best point ever, over all
points

[F_Global2, I_Global2] = optFun(F_Best2); % Value of best point
ever, over all points
X_Global2 = XX(:, I_Global2);%[2;-2]; % Best point ever, over all
points


%%%% Allocate memory for the dataLog
maxIter = options.maxIter;
dataLog(maxIter) = makeStruct(X,XX, V,V2, F,F2, X_Best, X_Best2,
F_Best, F_Best2, X_Global, X_Global2, F_Global,F_Global2,
I_Global,I_Global2);

batas1 = zeros(maxIter);
batas2 = zeros(maxIter);
batas3 = zeros(maxIter);
batas4 = zeros(maxIter);
batas5 = zeros(maxIter);
batas6 = zeros(maxIter);
batas7 = zeros(maxIter);
batas8 = zeros(maxIter);
batas9 = zeros(maxIter);
```

```
batas10 = zeros(maxIter);
batastot = zeros(maxIter);

SimpanX_Global_X = zeros(maxIter);
SimpanX_Global_Y = zeros(maxIter);
SimpanX_Global2_X = zeros(maxIter);
SimpanX_Global2_Y = zeros(maxIter);

info.X_Global = zeros(n,maxIter);
info.X_Global2 = zeros(n,maxIter);
info.F_Global = zeros(1,maxIter);
info.F_Global2 = zeros(1,maxIter);
info.I_Global = zeros(1,maxIter);
info.I_Global2 = zeros(1,maxIter);
info.X_Best_Var = zeros(n,maxIter);
info.F_Best_Var = zeros(1,maxIter);
info.X_Best_Mean = zeros(n,maxIter);
info.F_Best_Mean = zeros(1,maxIter);
info.X_Var = zeros(n,maxIter);
info.X_Var2 = zeros(n,maxIter);
info.F_Var = zeros(1,maxIter);
info.X_Mean = zeros(n,maxIter);
info.F_Mean = zeros(1,maxIter);
info.iter = 1:maxIter;

%%%% MAIN LOOP:
info.exitFlag = 1;   %Assume that we will reach maximum iteration
for iter = 1:maxIter
    r1 = rand(n,m);
%%% Compute new generation of points:
if iter > 1   % Then do an update on each particle

%yang depannya 1-->x
%yang depannya 2-->y belakang menunjukan partikel ke--
```

//digunakana untuk mengetahui posisi dari koordinat x dan y setiap quadrotot selanjutnya…x_best(n,m)…n->1 =➔ sumbu x, n->2 sumbu y

```
        X_Best(1,1)=X_Best(1,1)-((simpandata1)/10);
        X_Best(2,1)=X_Best(2,1)-((simpandata2)/10);

        X_Best(1,2)=X_Best(1,2)-(simpandata3/10);
        X_Best(2,2)=X_Best(2,2)-(simpandata4/10);

        X_Best(1,3)=X_Best(1,3)-(simpandata5/10);
        X_Best(2,3)=X_Best(2,3)-(simpandata6/10);

        X_Best(1,4)=X_Best(1,4)-(simpandata7/10);
        X_Best(2,4)=X_Best(2,4)-(simpandata8/10);

        X_Best(1,5)=X_Best(1,5)-(simpandata9/10);
        X_Best(2,5)=X_Best(2,5)-(simpandata10/10);
```

```
end
```

```
//digunakana untuk menyimpan posisi dari koordinat x
dan y setiap quadrotot selanjutnya…x_best(n,m)…n->1 =➔
sumbu x, n->2 sumbu y
```

```
    SimpanX_Global_1(iter,1) = X_Best(1,1);%x
    SimpanX_Global_1(iter,2) = X_Best(2,1); %y

    SimpanX_Global_2(iter,1) = X_Best(1,2);
    SimpanX_Global_2(iter,2) = X_Best(2,2);

    SimpanX_Global_3(iter,1) = X_Best(1,3);
    SimpanX_Global_3(iter,2) = X_Best(2,3);

    SimpanX_Global_4(iter,1) = X_Best(1,4);
    SimpanX_Global_4(iter,2) = X_Best(2,4);

    SimpanX_Global_5(iter,1) = X_Best(1,5);
    SimpanX_Global_5(iter,2) = X_Best(2,5);

if iter>1
        line([SimpanX_Global_1(iter-1,1)
SimpanX_Global_1(iter,1)],[SimpanX_Global_1(iter-1,2)
SimpanX_Global_1(iter,2)]);
        line([SimpanX_Global_2(iter-1,1)
SimpanX_Global_2(iter,1)],[SimpanX_Global_2(iter-1,2)
SimpanX_Global_2(iter,2)]);
        line([SimpanX_Global_3(iter-1,1)
SimpanX_Global_3(iter,1)],[SimpanX_Global_3(iter-1,2)
SimpanX_Global_3(iter,2)]);
        line([SimpanX_Global_4(iter-1,1)
SimpanX_Global_4(iter,1)],[SimpanX_Global_4(iter-1,2)
SimpanX_Global_4(iter,2)]);
        line([SimpanX_Global_5(iter-1,1)
SimpanX_Global_5(iter,1)],[SimpanX_Global_5(iter-1,2)
SimpanX_Global_5(iter,2)]);
end
```

```
    dataLog(iter) = makeStruct(X,XX, V,V2, F,F2, X_Best,X_Best2,
F_Best,F_Best2, X_Global,X_Global2, F_Global,F_Global2,
I_Global,I_Global2);
    info.X_Global(:,iter) = X_Global; %posisi Q1 tiap iterasi
    info.X_Global2(:,iter) = X_Global2; %posisi Q2 tiap iterasi
    info.F_Global(iter) = F_Global;
    info.F_Global2(iter) = F_Global2;
    info.I_Global(iter) = I_Global;
    info.I_Global2(iter) = I_Global2;
    info.X_Var(:,iter) = var(X, 0, 2);
    info.X_Best_Var(:,iter) = var(X_Best, 0, 2);
    info.X_Mean(:,iter) = mean(X, 2);
    info.X_Best_Mean(:,iter) = mean(X_Best, 2);
    info.F_Var(1,iter) = var(F);
    info.F_Best_Var(1,iter) = var(F_Best);
```

```
    info.F_Mean(1,iter) = mean(F);
    info.F_Best_Mean(1,iter) = mean(F_Best);

%%% Plot
if ~isempty(options.plotFun)
        options.plotFun(dataLog(iter), iter);
end

%%% Print:
    xVar = norm(info.X_Var(:,iter));
if strcmp('iter',options.display)
if mod(iter-1,options.printMod)==0
           fprintf('iter: %3d,   fBest: %9.3e,   fVar: %9.3e
xVar: %9.3e  \n',...
               iter, info.F_Global(iter), info.F_Var(1,iter),
xVar);
end
end
```

//digunakana untuk batas setiap quadrotor agar tidak tertabrak

```
    batas1(iter) =sqrt(((X_Best(1,1)-X_Best(1,2))*(X_Best(1,1)-
X_Best(1,2)))+((X_Best(2,1)-X_Best(2,2))*(X_Best(2,1)-
X_Best(2,2))));
    batas2(iter) =sqrt(((X_Best(1,1)-X_Best(1,3))*(X_Best(1,1)-
X_Best(1,3)))+((X_Best(2,1)-X_Best(2,3))*(X_Best(2,1)-
X_Best(2,3))));
    batas3(iter) =sqrt(((X_Best(1,1)-X_Best(1,4))*(X_Best(1,1)-
X_Best(1,4)))+((X_Best(2,1)-X_Best(2,4))*(X_Best(2,1)-
X_Best(2,4))));
    batas4(iter) =sqrt(((X_Best(1,1)-X_Best(1,5))*(X_Best(1,1)-
X_Best(1,5)))+((X_Best(2,1)-X_Best(2,5))*(X_Best(2,1)-
X_Best(2,5))));

    batas5(iter) =sqrt(((X_Best(1,2)-X_Best(1,3))*(X_Best(1,2)-
X_Best(1,3)))+((X_Best(2,2)-X_Best(2,3))*(X_Best(2,2)-
X_Best(2,3))));
    batas6(iter) =sqrt(((X_Best(1,2)-X_Best(1,4))*(X_Best(1,2)-
X_Best(1,4)))+((X_Best(2,2)-X_Best(2,4))*(X_Best(2,2)-
X_Best(2,4))));
    batas7(iter) =sqrt(((X_Best(1,2)-X_Best(1,5))*(X_Best(1,2)-
X_Best(1,5)))+((X_Best(2,2)-X_Best(2,5))*(X_Best(2,2)-
X_Best(2,5))));

    batas8(iter) =sqrt(((X_Best(1,3)-X_Best(1,4))*(X_Best(1,3)-
X_Best(1,4)))+((X_Best(2,3)-X_Best(2,4))*(X_Best(2,3)-
X_Best(2,4))));
    batas9(iter) =sqrt(((X_Best(1,3)-X_Best(1,5))*(X_Best(1,3)-
X_Best(1,5)))+((X_Best(2,3)-X_Best(2,5))*(X_Best(2,3)-
X_Best(2,5))));

    batas10(iter)=sqrt(((X_Best(1,4)-X_Best(1,5))*(X_Best(1,4)-
X_Best(1,5)))+((X_Best(2,4)-X_Best(2,5))*(X_Best(2,4)-
X_Best(2,5))));
```

```
    batastot(iter)=
(batas1(iter)+batas2(iter)+batas3(iter)+batas4(iter)+batas5(iter)+
batas6(iter)+batas7(iter)+batas8(iter)+batas9(iter)+batas10(iter))
/10;


if info.F_Var(1,iter) < options.tolFun
        info.exitFlag = 0;
        dataLog = dataLog(1:iter);
        info = truncateInfo(info,maxIter,iter);
break


elseif xVar < options.tolX
        info.exitFlag = 2;
        dataLog = dataLog(1:iter);
        info = truncateInfo(info,maxIter,iter);
break
%===batasan jk quadrotor jaraknya 0.5====
elseif batastot(iter) < 2
            info.exitFlag = 2;
            dataLog = dataLog(1:iter);
            info = truncateInfo(info,maxIter,iter);
break
%end
end
end

xBest = info.X_Global(:,end);
fBest = info.F_Global(end);
info.input = makeStruct(objFun, x0, xLow, xUpp, options);  %Copy
inputs
info.fEvalCount = iter*m;

%%% Print:
if strcmp('iter',options.display) ||
strcmp('final',options.display)
switch info.exitFlag
case 0
            fprintf('Optimization Converged. Exit: fVar < tolFun
\n');
case 1
            fprintf('Maximum iteration reached. \n');
case 2
            fprintf('Optimization Converged. Exit: norm(xVar) <
tolX \n');
end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
```

```
function info = truncateInfo(info,maxIter,iter)
%
% Removes the empty entries in the info struct

names = fieldnames(info);
for i=1:length(names)
if (isnumeric(info.(names{i})))   % Check if it's a matrix
if size(info.(names{i}),2) == maxIter    % Check if it is
iteration data
            info.(names{i}) = info.(names{i})(:,1:iter);
end
end
end
end
```

BIODATA

Albert Sudaryanto dilahirkan di Kediri pada Tanggal 13 Oktober 1990. Putra Pertama dari Bapak Budi Sudaryanto dan Ibu Endang Titik Subekti. Penulis menempuh pendidikan tingginya di Jurusan Teknik Elektronika, Politeknik Elektronika Negeri Surabaya pada tahun 2009. Pada bulan September 2013, penulis berhasil menyelesaikan pendidikan sarjananya dan mendapatkan gelar Sarjana Sains Terapan. Pada bulan September tahun 2015, Penulis mendapatkan beasiswa S2 BPPDN-PTN Baru dari DIKTI atas rekomendasi dari Politeknik Negeri Madiun untuk melanjutkan pendidikan magister di Jurusan Teknik Elektro, Institut teknologi Sepuluh Nopember (ITS) Surabaya dengan Bidang Keahlian Teknik Sistem Pengaturan. Penulis menyelesaikan studi magisternya pada bulan Maret 2018.